Honeywell

SERIES 6000

GENERAL MACRO ASSEMBLY PROGRAMMING (GMAP)

COURSE CODE 605

STUDENT HANDBOOK

Order Number: AF52, Rev. 1
<table>
<thead>
<tr>
<th>HANDLET REFERENCE NO.</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>General Course Outline</td>
<td>1</td>
</tr>
<tr>
<td>1-2</td>
<td>Course Topic Map</td>
<td>8</td>
</tr>
<tr>
<td>2-1</td>
<td>Series 6000 Processor</td>
<td>9</td>
</tr>
<tr>
<td>2-2</td>
<td>Basic System</td>
<td>10</td>
</tr>
<tr>
<td>2-3</td>
<td>Functional Modularity</td>
<td>11</td>
</tr>
<tr>
<td>2-4</td>
<td>Hardware Components</td>
<td>12</td>
</tr>
<tr>
<td>2-5</td>
<td>Multidimensional System</td>
<td>13</td>
</tr>
<tr>
<td>2-6</td>
<td>Multiprocessor System</td>
<td>14</td>
</tr>
<tr>
<td>2-7</td>
<td>Peripheral Subsystems</td>
<td>15</td>
</tr>
<tr>
<td>2-8</td>
<td>H-6000 Hardware Components</td>
<td>16</td>
</tr>
<tr>
<td>2-9</td>
<td>Multiprocessor System</td>
<td>17</td>
</tr>
<tr>
<td>2-10</td>
<td>Series 6000 Characteristics</td>
<td>18</td>
</tr>
<tr>
<td>2-11</td>
<td>Program Accessible Registers</td>
<td>19</td>
</tr>
<tr>
<td>2-12</td>
<td>Processor Indicators</td>
<td>20</td>
</tr>
<tr>
<td>2-13</td>
<td>Software System Major Components</td>
<td>21</td>
</tr>
<tr>
<td>2-14</td>
<td>System Software--GCOS</td>
<td>22</td>
</tr>
<tr>
<td>2-15</td>
<td>Other System Software</td>
<td>23</td>
</tr>
<tr>
<td>2-16</td>
<td>Language Processors</td>
<td>24</td>
</tr>
<tr>
<td>2-17</td>
<td>Time-Sharing System</td>
<td>25</td>
</tr>
<tr>
<td>2-18</td>
<td>Application Software</td>
<td>25.1</td>
</tr>
<tr>
<td>3-1</td>
<td>Coding Form</td>
<td>26</td>
</tr>
<tr>
<td>3-2</td>
<td>GMAP Compile (Single Assembly)</td>
<td>27</td>
</tr>
<tr>
<td>HANDOUT REFERENCE NO.</td>
<td>TITLE</td>
<td>PAGE</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3-3</td>
<td>GMAP Compile (Multiple Assemblies)</td>
<td>28</td>
</tr>
<tr>
<td>3-4</td>
<td>GMAP Card Deck Arrangement</td>
<td>29</td>
</tr>
<tr>
<td>3-5</td>
<td>Multiple GMAP Assembly - Card Deck</td>
<td>30</td>
</tr>
<tr>
<td>3-6</td>
<td>GMAP Control Records - Compile and Execute</td>
<td>31</td>
</tr>
<tr>
<td>3-7</td>
<td>Executing Object Programs</td>
<td>32</td>
</tr>
<tr>
<td>3-8</td>
<td>Compile and Execute with Object Decks</td>
<td>33</td>
</tr>
<tr>
<td>3-9</td>
<td>Compile and Execute - Deck Example</td>
<td>34</td>
</tr>
<tr>
<td>4-1</td>
<td>Word Formats</td>
<td>35</td>
</tr>
<tr>
<td>4-2</td>
<td>Single/Double Precision</td>
<td>36</td>
</tr>
<tr>
<td>4-3</td>
<td>Instruction Precision</td>
<td>37</td>
</tr>
<tr>
<td>4-4</td>
<td>Functional Groups of Pseudo-Operations</td>
<td>38</td>
</tr>
<tr>
<td>4-5</td>
<td>Storage Allocation Pseudo Operations</td>
<td>39</td>
</tr>
<tr>
<td>4-6</td>
<td>Binary Floating Point</td>
<td>40</td>
</tr>
<tr>
<td>4-7</td>
<td>Number Conversion Technique</td>
<td>41</td>
</tr>
<tr>
<td>5-1</td>
<td>Types of Instructions</td>
<td>43</td>
</tr>
<tr>
<td>5-2</td>
<td>Load and Store</td>
<td>44</td>
</tr>
<tr>
<td>5-2.1</td>
<td>Load/Store Sample Coding</td>
<td>44.1</td>
</tr>
<tr>
<td>5-2.2</td>
<td>LDAC Instruction</td>
<td>44.2</td>
</tr>
<tr>
<td>5-3</td>
<td>Address Modification</td>
<td>45</td>
</tr>
<tr>
<td>5-4</td>
<td>Indexing</td>
<td>46</td>
</tr>
<tr>
<td>5-5</td>
<td>Three Phases of an Instruction</td>
<td>47</td>
</tr>
<tr>
<td>5-6</td>
<td>Direct Operand</td>
<td>48</td>
</tr>
<tr>
<td>5-7</td>
<td>Effective Address</td>
<td>49</td>
</tr>
<tr>
<td>HANDOUT REFERENCE NO.</td>
<td>TITLE</td>
<td>PAGE</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>6-1</td>
<td>Literals and Data Generating Pseudo Ops</td>
<td>50</td>
</tr>
<tr>
<td>6-2</td>
<td>Decimal Literals</td>
<td>51</td>
</tr>
<tr>
<td>6-3</td>
<td>Decimal Values</td>
<td>52</td>
</tr>
<tr>
<td>6-4</td>
<td>Floating Point Values</td>
<td>53</td>
</tr>
<tr>
<td>6-5</td>
<td>Octal Literals</td>
<td>54</td>
</tr>
<tr>
<td>6-6</td>
<td>Octal Values</td>
<td>55</td>
</tr>
<tr>
<td>6-7</td>
<td>Hollerith Data</td>
<td>56</td>
</tr>
<tr>
<td>6-8</td>
<td>Instructional Literal</td>
<td>57</td>
</tr>
<tr>
<td>6-9</td>
<td>ARG Pseudo Operation</td>
<td>58</td>
</tr>
<tr>
<td>6-10</td>
<td>Variable Field Definition</td>
<td>59</td>
</tr>
<tr>
<td>6-11</td>
<td>Literals and Direct Operands</td>
<td>60</td>
</tr>
<tr>
<td>6-12</td>
<td>Hollerith Literal and DU</td>
<td>61</td>
</tr>
<tr>
<td>6-13</td>
<td>Floating Point Literal (DU)</td>
<td>62</td>
</tr>
<tr>
<td>6-14</td>
<td>Fixed Point Literal (DU)</td>
<td>63</td>
</tr>
<tr>
<td>6-15</td>
<td>Fixed Point Literal (Example 2)</td>
<td>64</td>
</tr>
<tr>
<td>7-1</td>
<td>Store Character (6-bit)</td>
<td>65</td>
</tr>
<tr>
<td>7-2</td>
<td>Store Byte (9-bit)</td>
<td>66</td>
</tr>
<tr>
<td>7-3</td>
<td>STCl and STC2</td>
<td>67</td>
</tr>
<tr>
<td>7-4</td>
<td>Special Load/Store</td>
<td>68</td>
</tr>
<tr>
<td>7-5</td>
<td>Shift Instructions</td>
<td>69</td>
</tr>
<tr>
<td>7-6</td>
<td>Fixed Point Arithmetic</td>
<td>70</td>
</tr>
<tr>
<td>7-7</td>
<td>Logical Add/Subtract</td>
<td>71</td>
</tr>
<tr>
<td>HANDOUT REFERENCE NO.</td>
<td>TITLE</td>
<td>PAGE</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>7-8</td>
<td>Carry and Overflow Indicators</td>
<td>72</td>
</tr>
<tr>
<td>7-9</td>
<td>Special Add/Subtract</td>
<td>73</td>
</tr>
<tr>
<td>7-10</td>
<td>Multiply/Divide</td>
<td>74</td>
</tr>
<tr>
<td>7-11</td>
<td>Transfer of Control</td>
<td>75</td>
</tr>
<tr>
<td>8-1</td>
<td>Assembly Listing</td>
<td>76</td>
</tr>
<tr>
<td>8-2</td>
<td>Saving Instruction Counter in Index Register</td>
<td>83.1</td>
</tr>
<tr>
<td>8-3</td>
<td>Saving Instruction Counter in Memory</td>
<td>83.2</td>
</tr>
<tr>
<td>8-4</td>
<td>Saving Instruction Counter and Indicators (STCl)</td>
<td>83.3</td>
</tr>
<tr>
<td>8-5</td>
<td>Saving Instruction Counter and Indicators (STI)</td>
<td>83.4</td>
</tr>
<tr>
<td>8-6</td>
<td>Saving Registers, Indicators and Instruction Counter</td>
<td>83.5</td>
</tr>
<tr>
<td>8-7</td>
<td>CALL Pseudo Operation</td>
<td>83.6</td>
</tr>
<tr>
<td>8-8</td>
<td>CALL Pseudo Operation - Expanded Version</td>
<td>83.7</td>
</tr>
<tr>
<td>8-9</td>
<td>SAVE Pseudo Operation - Expanded Version</td>
<td>83.8</td>
</tr>
<tr>
<td>8-10</td>
<td>CALL, SAVE and RETURN Pseudo Ops</td>
<td>83.9</td>
</tr>
<tr>
<td>8-11</td>
<td>CALL, SAVE, RETURN Coding Example</td>
<td>83.10</td>
</tr>
<tr>
<td>9-1</td>
<td>Compare Logic Flow</td>
<td>84</td>
</tr>
<tr>
<td>9-2</td>
<td>Positive Numbers &amp; Comparison Indicators</td>
<td>85</td>
</tr>
<tr>
<td>9-3</td>
<td>Negative Numbers &amp; Comparison Indicators</td>
<td>86</td>
</tr>
<tr>
<td>9-4</td>
<td>Comparison Indicators - Positive Register - Negative Memory</td>
<td>87</td>
</tr>
<tr>
<td>9-5</td>
<td>Algebraic and Logical Comparison</td>
<td>88</td>
</tr>
<tr>
<td>9-6</td>
<td>Fixed Point Compare</td>
<td>89</td>
</tr>
<tr>
<td>HANDOUT REFERENCE NO.</td>
<td>TITLE</td>
<td>PAGE</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>9-7</td>
<td>Special Purpose Compares</td>
<td>90</td>
</tr>
<tr>
<td>9-8</td>
<td>Boolean Operations Truth Table</td>
<td>91</td>
</tr>
<tr>
<td>9-9</td>
<td>Example of AND</td>
<td>92</td>
</tr>
<tr>
<td>9-10</td>
<td>Example of OR</td>
<td>93</td>
</tr>
<tr>
<td>9-11</td>
<td>Example of Exclusive OR</td>
<td>94</td>
</tr>
<tr>
<td>9-12</td>
<td>Boolean Compare Operations</td>
<td>95</td>
</tr>
<tr>
<td>9-13</td>
<td>Comparative AND</td>
<td>96</td>
</tr>
<tr>
<td>9-14</td>
<td>Comparative NOT AND</td>
<td>97</td>
</tr>
<tr>
<td>9-15</td>
<td>Example of Boolean Operations and Boolean Compares</td>
<td>98</td>
</tr>
<tr>
<td>10-1</td>
<td>Indirect then Tally Modification</td>
<td>99</td>
</tr>
<tr>
<td>10-2</td>
<td>Indirection (I) Example</td>
<td>100</td>
</tr>
<tr>
<td>10-3</td>
<td>Indirect Diagram</td>
<td>101</td>
</tr>
<tr>
<td>10-4</td>
<td>Indirect Variation Flowchart</td>
<td>102</td>
</tr>
<tr>
<td>10-4.1</td>
<td>Tally Word Formats</td>
<td>102.1</td>
</tr>
<tr>
<td>10-5</td>
<td>Address Tally Pseudo-Operations</td>
<td>103</td>
</tr>
<tr>
<td>10-6</td>
<td>Increment Address Decrement Tally Diagram</td>
<td>104</td>
</tr>
<tr>
<td>10-7</td>
<td>Increment Address, Decrement Tally Flowchart</td>
<td>105</td>
</tr>
<tr>
<td>10-8</td>
<td>Indexing Example</td>
<td>106</td>
</tr>
<tr>
<td>10-9</td>
<td>I and ID Example</td>
<td>107</td>
</tr>
<tr>
<td>10-10</td>
<td>Tally Refreshing</td>
<td>108</td>
</tr>
<tr>
<td>10-11</td>
<td>Decrement Address, Increment Tally Diagram</td>
<td>109</td>
</tr>
<tr>
<td>10-12</td>
<td>Decrement Address, Increment Tally Flowchart</td>
<td>110</td>
</tr>
<tr>
<td>10-13</td>
<td>DI Example</td>
<td>111</td>
</tr>
<tr>
<td>HANDOUT REFERENCE NO.</td>
<td>TITLE</td>
<td>PAGE</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>10-14</td>
<td>I, ID, and DI Example</td>
<td>112</td>
</tr>
<tr>
<td>10-15</td>
<td>Add Delta (AD) Diagram</td>
<td>113</td>
</tr>
<tr>
<td>10-16</td>
<td>Add Delta (AD) Flowchart</td>
<td>114</td>
</tr>
<tr>
<td>10-17</td>
<td>Subtract Delta (SD) Diagram</td>
<td>115</td>
</tr>
<tr>
<td>10-18</td>
<td>Subtract Delta (SD) Flowchart</td>
<td>116</td>
</tr>
<tr>
<td>10-19</td>
<td>AD and SD Example</td>
<td>117</td>
</tr>
<tr>
<td>10-20</td>
<td>Character Operations (6-bit) to Register</td>
<td>118</td>
</tr>
<tr>
<td>10-21</td>
<td>Character Operations (9-bit) to Register</td>
<td>119</td>
</tr>
<tr>
<td>10-22</td>
<td>Character Operations (6-bit) to Memory</td>
<td>120</td>
</tr>
<tr>
<td>10-23</td>
<td>Character from Indirect (CI) Diagram</td>
<td>121</td>
</tr>
<tr>
<td>10-24</td>
<td>Character from Indirect (CI) Flowchart</td>
<td>122</td>
</tr>
<tr>
<td>10-25</td>
<td>Sequence Character (SC) Flowchart</td>
<td>123</td>
</tr>
<tr>
<td>10-25.1</td>
<td>Sequence Character Reverse (SCR) Flowchart</td>
<td>123.1</td>
</tr>
<tr>
<td>10-25.2</td>
<td>Reversing Table -- Character by Character</td>
<td>123.2</td>
</tr>
<tr>
<td>10-26</td>
<td>CI and SC Example</td>
<td>124</td>
</tr>
<tr>
<td>10-27</td>
<td>CI and SC Example No. 2</td>
<td>125</td>
</tr>
<tr>
<td>10-28</td>
<td>Check Protection and Character Insertion</td>
<td>126</td>
</tr>
<tr>
<td>10-29</td>
<td>IDC Diagram</td>
<td>127</td>
</tr>
<tr>
<td>10-30</td>
<td>DIC Diagram</td>
<td>128</td>
</tr>
<tr>
<td>11-1</td>
<td>Repeat Instruction Format</td>
<td>129</td>
</tr>
<tr>
<td>11-2</td>
<td>Repeat Exit Conditions</td>
<td>130</td>
</tr>
<tr>
<td>11-3</td>
<td>Rules for Repeat Instructions</td>
<td>131</td>
</tr>
<tr>
<td>HANDOUT REFERENCE NO.</td>
<td>TITLE</td>
<td>PAGE</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>11-4</td>
<td>Address Modification of Repeated Instruction</td>
<td>132</td>
</tr>
<tr>
<td>11-5</td>
<td>Coding Example Using Repeat</td>
<td>133</td>
</tr>
<tr>
<td>11-6</td>
<td>RPTX and RPDX Diagram</td>
<td>134</td>
</tr>
<tr>
<td>11-7</td>
<td>RPTX Coding Examples</td>
<td>135</td>
</tr>
<tr>
<td>11-8</td>
<td>Repeat Double Format</td>
<td>136</td>
</tr>
<tr>
<td>11-9</td>
<td>Repeat Double Examples</td>
<td>137</td>
</tr>
<tr>
<td>11-10</td>
<td>Repeat Link Format and Execution</td>
<td>138</td>
</tr>
<tr>
<td>11-11</td>
<td>Key Words and Tables</td>
<td>139</td>
</tr>
<tr>
<td>11-11.1</td>
<td>RPL Coding Example</td>
<td>139.1</td>
</tr>
<tr>
<td>11-12</td>
<td>Execution of Repeat Link</td>
<td>140</td>
</tr>
<tr>
<td>11-13</td>
<td>BCD Instruction Flowchart</td>
<td>141</td>
</tr>
<tr>
<td>11-14</td>
<td>Binary to BCD Conversion Constants</td>
<td>142</td>
</tr>
<tr>
<td>11-15</td>
<td>BCD and A &amp; Q Registers</td>
<td>143</td>
</tr>
<tr>
<td>11-16</td>
<td>Converting Values to 6 Digits</td>
<td>144</td>
</tr>
<tr>
<td>11-17</td>
<td>Converting Maximum Values</td>
<td>145</td>
</tr>
<tr>
<td>11-18</td>
<td>XED Coding Example</td>
<td>146</td>
</tr>
<tr>
<td>12-1</td>
<td>Examples of Pseudo Operations</td>
<td>147</td>
</tr>
<tr>
<td>13-1</td>
<td>Macro Prototype and Expansion Call</td>
<td>160</td>
</tr>
<tr>
<td>13-2</td>
<td>Macro with Substitutable Arguments</td>
<td>161</td>
</tr>
<tr>
<td>13-3</td>
<td>Refreshing Tally Words with Macro Call</td>
<td>162</td>
</tr>
<tr>
<td>13-4</td>
<td>Location Field Symbols</td>
<td>163</td>
</tr>
<tr>
<td>13-5</td>
<td>Nesting of Macros</td>
<td>164</td>
</tr>
<tr>
<td>HANDOUT REFERENCE NO.</td>
<td>TITLE</td>
<td>PAGE</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>13-6</td>
<td>Macro with Conditional Pseudo Op</td>
<td>165</td>
</tr>
<tr>
<td>13-7</td>
<td>Macro with Indefinite Repeat</td>
<td>166</td>
</tr>
<tr>
<td>13-8</td>
<td>Macro with Set and Indefinite Repeat</td>
<td>167</td>
</tr>
<tr>
<td>14-1</td>
<td>External Interrupts</td>
<td>168</td>
</tr>
<tr>
<td>14-2</td>
<td>Internal Interrupts</td>
<td>169</td>
</tr>
<tr>
<td>15-1</td>
<td>Levels of Input/Output</td>
<td>170</td>
</tr>
<tr>
<td>16-1</td>
<td>File and Record Control (GFRC) General Loader (GLOAD)</td>
<td>183</td>
</tr>
<tr>
<td>16-2</td>
<td>$ TAPE File Card</td>
<td>184</td>
</tr>
<tr>
<td>16-3</td>
<td>Mass Storage File Card</td>
<td>185</td>
</tr>
<tr>
<td>16-4</td>
<td>SYSOUT and DATA File Cards</td>
<td>186</td>
</tr>
<tr>
<td>16-5</td>
<td>Media Conversion</td>
<td>187</td>
</tr>
<tr>
<td>16-6</td>
<td>Logical and Physical Records</td>
<td>188</td>
</tr>
<tr>
<td>16-7</td>
<td>Variable Add Fixed Length Records</td>
<td>189</td>
</tr>
<tr>
<td>16-8</td>
<td>Partitioned Records</td>
<td>190</td>
</tr>
<tr>
<td>16-9</td>
<td>File Control Block Arguments</td>
<td>191</td>
</tr>
<tr>
<td>16-10</td>
<td>File Designator Word</td>
<td>192</td>
</tr>
<tr>
<td>16-11</td>
<td>Call OPEN and FILCB</td>
<td>193</td>
</tr>
<tr>
<td>16-12</td>
<td>Call CLOSE and FILCB</td>
<td>194</td>
</tr>
<tr>
<td>16-13</td>
<td>GET Logical Record</td>
<td>195</td>
</tr>
<tr>
<td>16-14</td>
<td>PUT (WRITE) Logical Record</td>
<td>196</td>
</tr>
<tr>
<td>16-15</td>
<td>Initializing Editing Parameters</td>
<td>197</td>
</tr>
<tr>
<td>16-16</td>
<td>Write (CALL/WREC) Logical Record</td>
<td>198</td>
</tr>
<tr>
<td>16-17</td>
<td>Write (CALL PRINT) Logical Record</td>
<td>199</td>
</tr>
<tr>
<td>16-18</td>
<td>Write (CALL PUNCH) Logical Record</td>
<td>200</td>
</tr>
<tr>
<td>HANDOUT REFERENCE NO.</td>
<td>TITLE</td>
<td>PAGE</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>16-19</td>
<td>Physical Record Processing - Read Sequential Files</td>
<td>201</td>
</tr>
<tr>
<td>16-20</td>
<td>Physical Record Processing - Read Random Files</td>
<td>202</td>
</tr>
<tr>
<td>16-21</td>
<td>Physical Record Processing - Write Sequential Files</td>
<td>203</td>
</tr>
<tr>
<td>16-22</td>
<td>Physical Record Processing - Write Random Files</td>
<td>204</td>
</tr>
<tr>
<td>16-23</td>
<td>File Control Block Contents</td>
<td>205</td>
</tr>
<tr>
<td>16-24</td>
<td>File and Record Control (GFRC) Examples - Program Listing</td>
<td>206</td>
</tr>
<tr>
<td>16-25</td>
<td>Floating Load/Store Operations</td>
<td>228</td>
</tr>
<tr>
<td>17-2</td>
<td>Floating Load/Store Mnemonics</td>
<td>229</td>
</tr>
<tr>
<td>17-3</td>
<td>Loading Exponent Register</td>
<td>230</td>
</tr>
<tr>
<td>17-4</td>
<td>Floating Point Arithmetic Mnemonics</td>
<td>231</td>
</tr>
<tr>
<td>17-5</td>
<td>Floating Point Multiply and Divide</td>
<td>232</td>
</tr>
<tr>
<td>17-6</td>
<td>Floating Point Compare Mnemonics</td>
<td>233</td>
</tr>
<tr>
<td>17-7</td>
<td>Examples of Floating Point Numbers in Memory</td>
<td>234</td>
</tr>
<tr>
<td>17-8</td>
<td>Examples of Negative Values in Memory</td>
<td>235</td>
</tr>
<tr>
<td>17-9</td>
<td>Mixed Values in Memory</td>
<td>236</td>
</tr>
<tr>
<td>19-1</td>
<td>Register then Indirect Flowchart</td>
<td>237</td>
</tr>
<tr>
<td>19-2</td>
<td>Register then Indirect Modification</td>
<td>238</td>
</tr>
<tr>
<td>19-3</td>
<td>RI Coding Examples</td>
<td>239</td>
</tr>
<tr>
<td>19-4</td>
<td>Summary of Register Indirect (RI)</td>
<td>240</td>
</tr>
<tr>
<td>19-5</td>
<td>Algorithms: Register then Indirect (RI)</td>
<td>241</td>
</tr>
<tr>
<td>19-6</td>
<td>Indirect then Register Flowchart</td>
<td>242</td>
</tr>
<tr>
<td>19-7</td>
<td>Indirect then Register Modification</td>
<td>243</td>
</tr>
<tr>
<td>HANDOUT REFERENCE NO.</td>
<td>TITLE</td>
<td>PAGE</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>19-8</td>
<td>Summary of Indirect Register (IR)</td>
<td>244</td>
</tr>
<tr>
<td>19-9</td>
<td>Algorithms: Indirect then Register (IR)</td>
<td>245</td>
</tr>
<tr>
<td>19-10</td>
<td>IR and RI Modification</td>
<td>246</td>
</tr>
<tr>
<td>19-11</td>
<td>Coding Examples 1 to 3</td>
<td>247</td>
</tr>
<tr>
<td>19-12</td>
<td>Coding Examples 4 to 6</td>
<td>248</td>
</tr>
<tr>
<td>19-13</td>
<td>Coding Examples 7 and 8</td>
<td>249</td>
</tr>
<tr>
<td>20-1</td>
<td>Fixed-Point Data Formats</td>
<td>250</td>
</tr>
<tr>
<td>20-2</td>
<td>Floating Point Data Formats</td>
<td>251</td>
</tr>
<tr>
<td>20-3</td>
<td>Alphanumeric Data Formats</td>
<td>252</td>
</tr>
<tr>
<td>20-4</td>
<td>Decimal Formats for EIS</td>
<td>252.1</td>
</tr>
<tr>
<td>20-5</td>
<td>Floating Point Decimal Formats</td>
<td>253</td>
</tr>
<tr>
<td>20-6</td>
<td>Decimal Sign Position Formats</td>
<td>253.1</td>
</tr>
<tr>
<td>20-7</td>
<td>Operation Code Map for EIS</td>
<td>253.2</td>
</tr>
<tr>
<td>20-8</td>
<td>Operation Code Map (cont.)</td>
<td>253.3</td>
</tr>
<tr>
<td>20-9</td>
<td>EIS Multi-word Instructions</td>
<td>253.4</td>
</tr>
<tr>
<td>20-10</td>
<td>Single Word Instructions</td>
<td>253.5</td>
</tr>
<tr>
<td>20-11</td>
<td>Micro Operations</td>
<td>253.6</td>
</tr>
<tr>
<td>20-12</td>
<td>Operand Descriptors</td>
<td>253.7</td>
</tr>
<tr>
<td>20-13</td>
<td>EIS Address Modification</td>
<td>253.8</td>
</tr>
<tr>
<td>20-14</td>
<td>Indirect Word</td>
<td>253.9</td>
</tr>
<tr>
<td>20-15</td>
<td>Address Registers</td>
<td>253.91</td>
</tr>
<tr>
<td>20-16</td>
<td>Address Modification Flowchart</td>
<td>253.92</td>
</tr>
<tr>
<td>HANDOUT</td>
<td>REFERENCE NO.</td>
<td>TITLE</td>
</tr>
<tr>
<td>----------</td>
<td>---------------</td>
<td>--------------------</td>
</tr>
<tr>
<td></td>
<td>21-1</td>
<td>Assembly Listing</td>
</tr>
<tr>
<td></td>
<td>21-2</td>
<td>Common System Files</td>
</tr>
</tbody>
</table>
Course Name: GMAP Programming
Course Code: 605

General Course Outline

I. INTRODUCTION
   A. Introduction of instructor, students, course
   B. Class Administration
   C. Course Outline

II. OVERVIEW OF 6000
   A. Hardware Overview
   B. Software Overview
   C. General Characteristics of Processor

III. PROGRAM ORGANIZATION
   A. GMAP Coding Form and Symbolic Card Format
   B. Programming Conventions
   C. Submitting GMAP Programs
   D. Typical Deck Setups
   E. Requirements for Submitting GMAP Programs

IV. DATA REPRESENTATION
   A. Internal Data Representation
   B. Processor Handling of Information
   C. Instruction Formats
   D. Data Word Formats
   E. Introduction to Pseudo Operations
      1. Definition
      2. Functional Groups
F. Storage Allocation Pseudo Operations
G. Data Generation Pseudo Operations
H. Summary

V. INTRODUCTION TO INSTRUCTIONS
A. Format
B. Types
C. Data Movement
D. Introduction to Address Modification
E. Using Address Modification
F. Direct Operands
G. Effective Address Instructions
H. Store Register Instructions

VI. LITERALS
A. Introduction
B. Decimal Literals
C. Octal Literals
D. Alphanumeric Literals
E. Instruction Literals
F. Variable Field Literals
G. Literals Modified by Direct Operand

VII. INSTRUCTION REPERTOIRE
A. Review
B. Instructions to be Discussed
C. Miscellaneous Store Instructions
D. Register Load and Store
E. Shift Instructions
F. Arithmetic Operations In 6000
G. Fixed Point Arithmetic
H. Fixed Point Instructions
I. Transfer of Control

VIII. LISTINGS AND LINKAGE
A. Contents
B. Introduction to Program Linkage
   1. Subroutines
   2. Subprograms
C. Program Linkage
D. CALL Pseudo Operation
E. SAVE and RETURN

IX. COMPARE AND LOGICAL INSTRUCTIONS
A. Introduction to Compare Instructions
B. Basic Compare Instructions
C. Boolean Functions
D. Boolean Analogy
E. Boolean Instructions
F. Other Boolean Operations
G. Comprehensive Example

X. TALLY WORDS AND ADDRESS MODIFICATION
A. Four Types of Modification
B. Indirect then Tally Variations
C. Indirect Addressing
D. Tally Pseudo Operations
E. Increment Address, Decrement Tally
F. Refreshing Tally Words
G. Decrement Address, Increment Tally
H. Add Delta and Subtract Delta Modification
I. Character Operations
J. Other Variations of IT Modification

XI. MISCELLANEOUS INSTRUCTIONS
A. Introduction
B. Repeat Instructions - Introduction
C. Repeat (RPT)
D. RPTX Instruction
E. Repeat Double (RPD)
F. Repeat Link (RPL-RLPX)
G. BCD Instruction
H. XEC and XED Instructions
I. Master Mode Entries

XII. PSEUDO OPERATIONS
A. General Description
B. Functional Groups
C. Control Pseudo Operations
D. Location Counter Pseudo Operations
E. Symbol-Defining Pseudo Operations
F. Data Generating Pseudo Ops
G. Storage Allocation
H. Conditional Pseudo Operations
I. Special Word Format
XIII. MACRO OPERATIONS
   A. Introduction
   B. Defining the Macro Prototype
   C. Using the Macro Prototype
   D. Arguments for Macros
   E. Created Symbols
   F. Nesting of Macros
   G. Pseudo Operations within Macros
   H. System Macros

XIV. EXTERNAL AND INTERNAL INTERRUPTS
   A. Execution of Interrupts
   B. Internal Interrupts
   C. Categories of Internal Interrupts
   D. Master Mode Entry Routines
   E. MME's for Activity Termination
   F. MME GESNAP

XV. INTRODUCTION TO INPUT/OUTPUT
   A. Levels of Input/Output in 6000
   B. Necessary Ingredients of I/O Coding
      1. Logical File Designator
      2. Input/Output Buffer
      3. Data Control List
      4. Input/Output Command

XVI. File and Record Control (GFRC)
   A. Introduction to Input/Output
   B. Introduction to File and Record Control (GFRC)
C. File Control Records
D. Standard System Formay
E. Logical Record Formats
F. File Control Block
G. File Designator Word
H. File and Record Control (GPRC) Calling Sequences
I. Functional Groups of Calling Sequences
J. Calling Sequences for File Preparation
K. Logical Record Processing
L. Input/Output Editor
M. Physical Record Processing

XVII. FLOATING POINT
A. Introduction
B. Purpose
C. Load and Store
D. Arithmetic Instructions
E. Miscellaneous Instructions
F. Compare Instructions
G. Summary of Arithmetic Operations

XVIII. REVIEW OF INSTRUCTIONS
A. Data Movement
B. Fixed-Point Arithmetic
C. Boolean Operations
D. Comparison
E. Transfer of Control
F. Miscellaneous Operations
XIX. ADDRESS MODIFICATION - REVIEW AND EXTENSION
   A. Review of Modification Capabilities
   B. Register Modification
   C. Indirect then Tally Modification
   D. Register then Indirect Modification
   E. Indirect then Register Modification
   F. Coding Examples

XX. EXTENDED INSTRUCTION SET
   A. Introduction
   B. Data Word Formats
   C. Instruction Word Formats
   D. Multi-word Instructions
   E. Single Word Instructions
   F. Micro Operations

XXI. ANALYSIS OF LISTINGS
   A. Review
   B. Files Produced by System Input (GIN)
   C. Accounting Information
      1. Job Information
      2. Activity Information
   D. Assembly Listing
   E. Memory Map
### Course Topic Map

#### Week 1

<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th>T</th>
<th>W</th>
<th>T</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.M.</td>
<td>Introduction</td>
<td>Data Representation</td>
<td>Instruction Repertoire</td>
<td>Compare and Logical Operations</td>
<td>Miscellaneous Instructions</td>
</tr>
<tr>
<td></td>
<td>Overview</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P.M.</td>
<td>Program</td>
<td>to Instructions</td>
<td>Listings and Linkage</td>
<td>Tally Words and Address Modification</td>
<td>Pseudo Operations</td>
</tr>
<tr>
<td></td>
<td>Organization</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>to Literals</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Week 2

<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th>T</th>
<th>W</th>
<th>T</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.M.</td>
<td>Macro Operations</td>
<td>File and Record Control (GFRC)</td>
<td>Floating Point</td>
<td>Extended Instruction Set</td>
<td>Final Exam</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Interrupts</td>
<td>Review of Instructions</td>
<td>Analysis of Listings</td>
<td>Review of Exam</td>
<td></td>
</tr>
<tr>
<td>P.M.</td>
<td>Introduction to Input/Output</td>
<td>Address Modification RI &amp; IR</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FUNCTIONAL MODULARITY

PORT CONNECTIONS

MEMORY MODULE

MEMORY MODULE

PROCESSOR

PROCESSOR

PROCESSOR

PROCESSOR

IOM

IOM

IOM OR COMMUNICATIONS PROCESSOR

IOM OR COMMUNICATIONS PROCESSOR

CONNECTIONS ARE MADE FROM EVERY MEMORY MODULE TO EVERY ACTIVE MODULE (IOM, PROCESSOR, OR COMMUNICATIONS PROCESSOR).
MULTIDIMENSIONAL SYSTEM

PROCESSOR

SYSTEM CONTROLLER

IOM

PERIPHERALS

DATANET 355

TERMINALS

65,536 → 65,536 → 65,536 → 65,536
SERIES 6000

PROCESSOR

64K

SYSTEM CONTROLLER

SYSTEM CONTROLLER

DATANET 355

REMOTE TERMINALS

IOM

64K

64K

CONSOLE

CONSOLE

READER

READER

PUNCH

PRINTER

PRINTER

PRINTER

PERIPHERAL SUBSYSTEMS
## SERIES 6000 CHARACTERISTICS

<table>
<thead>
<tr>
<th></th>
<th>Model 6030/6040</th>
<th>Model 6050/6060</th>
<th>Model 6070/6080</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Memory Size in Words (36-bit word)</strong></td>
<td>65,536 - 131,072</td>
<td>98,304 - 524,288</td>
<td>131,072 - 1,048,576</td>
</tr>
<tr>
<td><strong>Cycle Time (microseconds per two words)</strong></td>
<td>1.2</td>
<td>1.2</td>
<td>0.5</td>
</tr>
<tr>
<td><strong>No. of Data Channels</strong></td>
<td>16</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td><strong>Max. Transfer Rate Per IOM (chars/sec.)</strong></td>
<td>1.3M</td>
<td>3.7M</td>
<td>6.0M</td>
</tr>
<tr>
<td><strong>Peripheral Capacity (subsystems)</strong></td>
<td>16</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td><strong>Interleaving</strong></td>
<td>No</td>
<td>2&amp;4-way</td>
<td>2&amp;4-way</td>
</tr>
<tr>
<td><strong>Instruction Overlapping</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Instructions per Second (max.)</strong></td>
<td>340,000</td>
<td>550,000</td>
<td>1,400,000</td>
</tr>
</tbody>
</table>
PROGRAM ACCESSIBLE REGISTERS

0 35 A REGISTER

0 35 Q REGISTER

0 A 35 Q 71 ACCUMULATOR

0.7 E REGISTER (FLOATING POINT EXPONENTS)

0 E 8 A 35 EXP MANTISSA E/A REGISTER (FLOATING POINT)

0 E 8 A 35 Q 71 EXP MANTISSA E/A/Q REG.

0 23 26 TIMER REGISTER

0 17 INSTRUCTION COUNTER

0 17 INDEX REGISTERS (0-7)

0 8 17 BAR 0 0 WORD 17 19 23 CH BIT ADDR.

REG.
<table>
<thead>
<tr>
<th>NOT USED</th>
<th>INDICATOR BITS</th>
<th>NOT USED</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>18 ZERO</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>19 NEGATIVE</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>20 CARRY</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>21 OVERFLOW</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>22 EXP. OVERFLOW</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>23 EXP. UNDERFLOW</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>24 OVERFLOW MASK</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>25 TALLY RUNOUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>26 PARITY ERROR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>27 PARITY MASK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>28 MASTER MODE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>29 TRUNCATION (EIS)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30 MULTIWORD INSTRUCTION (EIS)</td>
<td>INTERRUPT</td>
</tr>
</tbody>
</table>
H-6000 SOFTWARE SYSTEM MAJOR COMPONENTS

OPERATING SYSTEM (GCOS)

OTHER SYSTEM SOFTWARE

LANGUAGE PROCESSORS

TIME-SHARING SYSTEM

APPLICATIONS SOFTWARE
SYSTEM SOFTWARE

OPERATING SYSTEM - GCOS

GENERAL COMPREHENSIVE OPERATING SUPERVISOR

GENERAL CAPABILITIES

MULTIDIMENSIONAL MODES OF PROCESSING
LOCAL BATCH PROCESSING
REMOTE BATCH PROCESSING
REMOTE ACCESS PROCESSING
TIME-SHARING
TRANSACTION PROCESSING

MULTIPROGRAMMING

MULTIPROCESSING

CENTRAL FILE SYSTEM

FUNCTIONAL CHARACTERISTICS

RESOURCE MANAGEMENT

JOB SCHEDULING

JOB PRIORITY ALLOCATION

DATA BASE MANAGEMENT

PROGRAM AND DATA FILE SECURITY CONTROL

ON-LINE SYSTEM DEVELOPMENT

EASE OF USE
SYSTEM SOFTWARE

GENERAL LOADER

FILE AND RECORD CONTROL (GFRC)

UTILITY PACKAGE

SYSTEM EDITOR

BULK MEDIA CONVERSION (BMC)

SYSTEM MEDIA CONVERSION

Input Media Conversion (GIN)

Output Media Conversion (SYSOUT)
LANGUAGE PROCESSORS

MACRO ASSEMBLER (GMAP)

FORTRAN COMPILER

COBOL COMPILER

ALGOL COMPILER

JOVIAL COMPILER

MISC. PROCESSORS

SORT/MERGE

IDS

ISP
TIME-SHARING SYSTEM

PROGRAMMING LANGUAGES
   BASIC
   FORTRAN
   ABACUS

REMOTE JOB ENTRY
   CARDIN
   JOUT
   SCAN
   RBUG

FILE EDITING, INSPECTION AND MAINTENANCE
   EDITOR
   RUNOFF
   DATA BASIC
   ACCESS
   IDS DATA QUERY

TIME-SHARING SYSTEM EXTENSION & MAINTENANCE
   LODX Subsystem
   TDS
   SABT
   MASTER

TIME-SHARING LIBRARY
   APPLICATION PROGRAMS
   FORTRAN LIBRARY GENERATOR AND EDITOR
   TSLG
   LIBED
APPLICATION SOFTWARE

LP6000 - LINEAR PROGRAMMING SYSTEM

APT - AUTOMATIC PROGRAMMED TOOLS

PERT/TIME - PROGRAM EVALUATION REVIEW TECHNIQUE/TIME

PERT/COST - PROGRAM EVALUATION REVIEW TECHNIQUE/COST

SIMSCRIPT - SIMULATION LANGUAGE

BMD - BIOMEDICAL STATISTICAL PROGRAMS

MATHPAC - MATHEMATICAL/ENGINEERING PROGRAMS

SERIES 6000 PARTS EXPLOSION SYSTEM (GEPEXS)

SERIES 6000 INVENTORY MANAGEMENT SYSTEM (GEIMS)

CEP - CIVIL ENGINEERING PACKAGE

SERIES 6000 TIME SERIES FORECASTING (GECAST)
**CODING FORM**

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>E/O</th>
<th>OPERATION</th>
<th>ADDR/MOD</th>
<th>COMMENTS</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>GO</td>
<td>E</td>
<td>LDA</td>
<td>WORD,7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 CHAR.</td>
<td>E</td>
<td>INST.</td>
<td>VARIABLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAX.</td>
<td>0</td>
<td>MNEMONICS,</td>
<td>LENGTH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(A-Z)</td>
<td>0</td>
<td>PSEUDO</td>
<td>FIELD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0-9)</td>
<td></td>
<td>Ops.</td>
<td>SUBFIELDS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(,)</td>
<td>8</td>
<td>MACRO</td>
<td>SEPARATED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LIST CHAR.</td>
<td></td>
<td>NAMES.</td>
<td>BY COMMAS.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MUST NOT</td>
<td>E</td>
<td>NO</td>
<td>ABSOLUTE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BE ZERO</td>
<td>0</td>
<td>SPECIAL</td>
<td>SYMBOLIC. OR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AND MUST</td>
<td></td>
<td>CHARS.</td>
<td>RELATIVE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HAVE AT</td>
<td>8</td>
<td>BLANKS</td>
<td>ADDRESSING.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEAST</td>
<td></td>
<td>PERMITTED</td>
<td>ALGEBRAIC-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ONE</td>
<td></td>
<td></td>
<td>BOOLEAN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NON-NUMERIC CHAR.</td>
<td></td>
<td></td>
<td>EXPRESSIONS.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OPERATORS.</td>
<td>* + - / ALSO =</td>
<td></td>
</tr>
</tbody>
</table>
GMAP COMPIL
(SINGLE GMAP ASSEMBLY)

1   8   16

$ SNUMB 34871
$ IDENT 1234,FIELDTRNG....
$ GMAP NDECK,COMDK

(SOURCE DECK)
.
.
.
.
.

$ ENDD

***EOF
GMAP COMPIL

(MULTIPLE GMAP ASSEMBLIES)

1 8 16

$ SNUMB 34872

$ IDENT 1234,FIELDTRNG,....

$ GMAP

(SOURCE DECK)

$ GMAP

(SOURCE DECK)

$ GMAP

(SOURCE DECK)

$ GMAP

(SOURCE DECK)

$ ENDJOB

***EOF
GMAP COMPIL

CARD DECK ARRANGEMENT

***EOF

$ ENDJOB

$ GMAP NDECK

$ GMAP NDECK

$ IDENT

$ SNUMB
MULTIPLE GMAP ASSEMBLY JOB

***EOF
$ENDJOB
$LIMITS
$EXECUTE
$SOURCE CARDS
$GMAP
$SOURCE CARDS
$GMAP
$SOURCE CARDS
$GMAP
$IDENT
$SNUMP
GMAP CONTROL RECORDS

(COMPILTE AND EXECUTE)

1 8 16

$ SNUMB 34873

$ IDENT 1234,FIELDTRNG,....

$ GMAP NDECK,COMDK

(SOURCE DECK)

$ EXECUTE

$ LIMITS 3.4K,1000

$ SYSOUT AL (PRINTER OUTPUT)

$ DATA IN

(DATA FOR EXECUTION)

$ ENDJOB

***EOF
EXECUTING OBJECT PROGRAMS

$ SNUMB  34874
$ IDENT  1234,FIELDTRNG,....
$ OBJECT

(BINARY DECK)

$ DKEND
$ OBJECT

(BINARY DECK)

$ DKEND
$ EXECUTE DUMP
$ LIMITS 4,6K,2000
$ SYSOUT X2 (PRINTER OUTPUT)
$ DATA IN

(DATA FOR OBJECT PROGRAM)

$ ENDJOB

***EOF
COMPILE AND EXECUTE
WITH OBJECT DECKS

1 8 16

$ SNUMB 34875
$ IDENT 1234, FIELDS, ...
$ GMAP

(SOURCE DECK)

$ OBJECT

(OBJECT DECK)

$ DKEND

$ EXECUTE

$ LIMITS 3.5K, 2500

$ SYSOUT B3 (PUNCH OUTPUT)

$ DATA

(DATA FOR OBJECT PROGRAM)

$ ENDJOB

***EOF
COMPILE AND EXECUTE WITH OBJECT DECK

***EOF

$ ENDJOB

$ LIMITS .5K, 1000

$ EXECUTE

$ GMAP

DECK

$ DKEND

$ OBJECT

$ IDENT

$ SNUMB
WORD FORMATS

BINARY

\[ 0 \ 1 \ 35 \]

\begin{array}{c}
S
\end{array}

BCD

\[
\begin{array}{cccccccc}
0 & 5 & 6 & 1112 & 17 & 18 & 23 & 24 & 29 & 30 & 35 \\
0 & 1 & 2 & 3 & 4 & 5
\end{array}
\]

9-BIT BYTE

\[
\begin{array}{cccccc}
0 & 8 & 9 & 17 & 18 & 26 & 27 & 35 \\
0 & 1 & 2 & 3
\end{array}
\]

INSTRUCTION

\[
\begin{array}{cccccccc}
0 & 17 & 18 & 26 & 28 & 30 & 35 \\
\text{OPERAND ADDR.} & \text{OPERATION CODE} & Z & I & Z & \text{TAG} & \text{TM} & \text{TD}
\end{array}
\]
SINGLE/DOUBLE PRECISION

SINGLE

DOUBLE OR Y-PAIR

HALF
INSTRUCTION PRECISION

PRECISION

HALF-WORD

UPPER HALF

LOWER HALF

SINGLE WORD

DOUBLE WORD

0 EVEN ADDRESS 35 36 ODD ADDRESS 71
FUNCTIONAL GROUPS
OF
PSEUDO OPERATIONS

CONTROL
LOCATION COUNTER
SYMBOL DEFINING
DATA GENERATING
STORAGE ALLOCATION
SPECIAL
MACRO
CONDITIONAL
PROGRAM LINKAGE
ADDRESS TALLY
MISCELLANEOUS
STORAGE ALLOCATION

PSEUDO-OPERATIONS

1 TABLE
8 BSS
16 1000

(TABLE) 2500

TABLA BFS 1000

STA TABLE

STA TABLA-1000 3499

STAQ TABLE

3500

STAQ TABLA-2

4499

* LDAQ TABLE+5

STAQ TABLA-2

4500

* Addressing Error
BINARY FLOATING POINT

SINGLE-WORD PRECISION

\[ \begin{array}{c}
0 & 1 & 7 & 8 & \text{S} & \text{S} & 35 \\
\hline
\text{E} & \text{M} \\
\end{array} \]

DOUBLE-WORD PRECISION

\[ \begin{array}{c}
0 & 1 & 7 & 8 & 9 & \text{S} & 71 \\
\hline
\text{E} & \text{M} \\
\end{array} \]

LOWEST PERMISSIBLE EXPONENT = -128

LARGEST POSSIBLE EXPONENT = +127
## Number Conversion Technique

### 1. Decimal

<table>
<thead>
<tr>
<th>11.054</th>
</tr>
</thead>
</table>

### 2. Octal Integer Conversion

<table>
<thead>
<tr>
<th>8</th>
<th>11</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 3. Octal Fraction Conversion

<table>
<thead>
<tr>
<th>0.054</th>
</tr>
</thead>
</table>

### 4. Octal

+13.033514

### 5. Binary Conversion

| 001011.000011011101001100 |

### 6. Floating Point

- S | Exponent | S | Mantissa |
- ---|----------|---|----------|
- 0 | 0000100  | 1911.000011011101001100 |

- Move four places left

### 7. Floating Point (Octal)

| 010541564600 |

---

STUDENT HANDBOOK REFERENCE E NO. 4-7
8. OCTAL INTEGER CONVERSION

\[
\begin{align*}
13_8 \\
\times 8 \\
8 \\
+ 3 \\
\Rightarrow 11_{10}
\end{align*}
\]

9. OCTAL FRACTION CONVERSION

\[
0.33514_8 \\
\times 12 \\
067230 \\
033514 \\
\Rightarrow 424370 \\
\times 12_8 \\
1050760 \\
424370 \\
\Rightarrow 314660 \\
\times 12_8 \\
631540 \\
314660 \\
\Rightarrow 000340
\]

10. DECIMAL (BCD)

\[11.054_{10}\]
1. DATA MOVEMENT
   LOAD STORE SHIFT
2. FIXED POINT ARITHMETIC
   ADD-SUBTRACT-MULTIPLY-DIVIDE
3. LOGICAL
   BOOLEAN (AND-OR-EXOR), COMPARE
4. TRANSFER OF CONTROL
   CONDITIONAL & UNCONDITIONAL
5. FLOATING POINT
   F. P. DATA MOVEMENT
   F. P. ARITHMETIC
   F. P. COMPARES
6. MASTER MODE
   CONNECT I/O, M. C. CMNDS., DIS
7. MISCELLANEOUS
   REPEATS, BCD, EFFECTIVE ADDRESS
## LOAD/STORE

<table>
<thead>
<tr>
<th>LDA</th>
<th>STA</th>
<th>LCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDQ</td>
<td>STQ</td>
<td>LCQ</td>
</tr>
<tr>
<td>LDAQ</td>
<td>STAQ</td>
<td>LCAQ</td>
</tr>
<tr>
<td>LDXₙ</td>
<td>STXₙ</td>
<td>LCXₙ</td>
</tr>
<tr>
<td>LXLₙ</td>
<td>SXLₙ</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ZERO</th>
<th>NONE</th>
<th>ZERO</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEG</td>
<td></td>
<td>NEG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>O·FLO</td>
</tr>
</tbody>
</table>
LOAD/STORE
SAMPLE CODING

1  8  16

R1  STA  B
    \    /
R4  LDQ  A
    \    /LDA \=6HKFLDKA
    \    /STAQ P1
    \    /STQ  P2
    \    /LDX7 R1
    \    /STX7 R4
    \    /LCAQ =5DOB71
    \    /STAQ C1
    \    /
    \    /
C1  EDEC  0.0
P1  BCI   2
P2  DEC   0
A   BSS   1
B   OCT   -777777777777

44.1
LDAC INSTRUCTION

(LDAC MEMORY)

BEFORE EXECUTION:

<table>
<thead>
<tr>
<th>MEMORY</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B C D E F</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A-REGISTER</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>X X X X X X</td>
<td></td>
</tr>
</tbody>
</table>

AFTER EXECUTION:

<table>
<thead>
<tr>
<th>MEMORY</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A-REGISTER</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B C D E F</td>
<td></td>
</tr>
</tbody>
</table>
### ADDRESS MODIFICATION

#### INSTRUCTION FORMAT:

<table>
<thead>
<tr>
<th>ADDRESS FIELD</th>
<th>OP CODE</th>
<th>TM</th>
<th>TD</th>
</tr>
</thead>
</table>

#### TM - TYPE OF MODIFICATION

- **00** - REGISTER \( R \)
- **01** - REGISTER THEN INDIRECT \( RI \)
- **11** - INDIRECT THEN REGISTER \( IR \)
- **10** - INDIRECT THEN TALLY \( IT \)

#### TD - REGISTER OR TALLY DESIGNATOR

\[ \{ \begin{array}{c} \text{REGISTER DESIGNATOR} \\ \text{REGISTER DESIGNATOR} \\ \text{TALLY DESIGNATOR} \end{array} \]
INDEXING

REG. SPECIFIED BY TD

PLUS

\[ y \quad \text{OP CODE} \quad \text{TAG} \]

\[ TM \quad TD \]

\[ = \quad y \quad (\text{EFFECTIVE ADDRESS}) \]

**ALGORITHM**

- \( Xn \ Y = y+C(Xn) \quad 0-17 \)
- \( AU \ Y = y+C(A) \quad 0-17 \)
- \( AL \ Y = y+C(A) \quad 18-35 \)
- \( QU \ Y = y+C(Q) \quad 0-17 \)
- \( QL \ Y = y+C(Q) \quad 18-35 \)
- \( IC \ Y = y+C(IC) \quad 0-17 \)

**CODING**

- \( Y,0-7 \)
- \( Y,AU \)
- \( Y,AL \)
- \( Y,QU \)
- \( Y,QL \)
- \( Y,IC \)
DIRECT OPERAND

0  17  18  35
A, Q OR A OR Q
INDEX REGS. REGISTERS ONLY

DU

DL

18-BIT OPERAND

Y OPERAND  OP CODE  TAG

ALGORITHM     CODING
DU  Y⇒C(R)  0-17      Y,DU
DL  Y⇒C(R)  18-35     Y,DL
EFFECTIVE ADDRESS

A REGISTER

1 2 3 4 5 6 0 0 0 0 0 0

INSTRUCTION WORD

1 2 3 4 5 6 E A A

ZERO'S

Q REGISTER

6 5 4 3 2 1 0 0 0 0 0 0

INSTRUCTION WORD

6 5 4 3 2 1 E A Q

INDEX REGISTER

2 5 2 5 2 5

INSTRUCTION WORD

2 5 2 5 2 5 E A X
<table>
<thead>
<tr>
<th>LITERALS</th>
<th>PSEUDO OPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECIMAL</td>
<td>DEC</td>
</tr>
<tr>
<td>OCTAL (=O)</td>
<td>OCT</td>
</tr>
<tr>
<td>HOLLERITH (=H)</td>
<td>BCI</td>
</tr>
<tr>
<td></td>
<td>ASCII</td>
</tr>
<tr>
<td></td>
<td>UASCi</td>
</tr>
<tr>
<td>INSTRUCTION (=M)</td>
<td>ARG</td>
</tr>
<tr>
<td>VARIABLE FIELD  (=V)</td>
<td>VFD</td>
</tr>
</tbody>
</table>

All literals must be preceded with an equal sign in column 16 of the variable field of the coding sheet.
DECIMAL LITERALS

INTEGERS:

DISTINGUISHED BY THE ABSENCE OF A DECIMAL POINT, THE LETTER B, THE LETTER E, OR THE LETTER D.

SINGLE PRECISION FLOATING POINT:

DISTINGUISHED BY THE PRESENCE OF A DECIMAL POINT, THE LETTER E, OR BOTH.

DOUBLE PRECISION FLOATING POINT:

DISTINGUISHED BY THE PRESENCE OF THE LETTER D.

FIXED POINT:

DISTINGUISHED BY THE PRESENCE OF THE LETTER B. (OVERRIDES E OR D)
DECIMAL VALUES

SINGLE PRECISION

DEC 5
LDA =5
DEC 5B35
LDA =5B35
DEC 5E0B35

DOUBLE PRECISION

DEC 0,5
DEC 5DOB71
LDAQ =5DOB71

SIGNED VALUES

DEC -5
LDA =-5

STUDENT HANDBOOK
REFERENCE NO. 6-3
FLOATING POINT VALUES

SINGLE PRECISION

DEC  5.
LDA  = 5.
DEC  5E0
LDA  = 5E0

DOUBLE PRECISION

DEC  5D0
LDAQ = 5D0

NEGATIVE VALUES

DEC  -5.
DEC  -5E0
LDA  = -5.
OCTAL LITERALS

The octal literal consists of the character $\emptyset$, followed by a signed or unsigned octal integer which may be from one (1) to twelve (12) digits in length plus a sign.

The assembler will store the integer in a word right-justified. The word will be stored in its real form and will not be complemented if there is a presence of a minus sign (-).
OCTAL VALUES

SINGLE PRECISION

OCT 5
LDA = 05

DOUBLE PRECISION

OCT 0.5

SIGNED VALUES

OCT -5
LDA = 0.5

100----0101
0 3 5
HOLLERITH DATA

SINGLE PRECISION

BCI  1, ABCDEF

A B C D E F
0  3
5

LDA  = 6HABCDEF

DOUBLE PRECISION

BCI  2, ABCDEFABCDEF

LDAQ  = 12HABCDEFABCDEF

A B C D E F A B C D E F
0  7
1

56
INSTRUCTION LITERAL

LDA = MTRA CAT,7

ADDR | OP | TAG
--- | --- | ---
CAT | TRA | 07

LDA = MLDA DOG,DU

ADDR | OP | TAG
--- | --- | ---
DOG | LDA | DU
ARG PSEUDO OPERATION

ARG 4

<table>
<thead>
<tr>
<th>ADDR</th>
<th>OP</th>
<th>TAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>000</td>
<td>00</td>
</tr>
</tbody>
</table>

ARG 5,4

<table>
<thead>
<tr>
<th>ADDR</th>
<th>OP</th>
<th>TAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>000</td>
<td>04</td>
</tr>
</tbody>
</table>

ARG CAT

<table>
<thead>
<tr>
<th>ADDR</th>
<th>OP</th>
<th>TAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAT</td>
<td>000</td>
<td>00</td>
</tr>
</tbody>
</table>
LITERALS MODIFIED BY DIRECT OPERANDS

LDA = 3HAAAA, DU  |  LDQ = 10, DU
LDQ = 3HQQQQ, DL  |  LXLN = 124, DL
FLD = 327D2, DU   |  LDXN = 1B18, DU
*LDA = 1B5, DU

ASSEMBLY TIME

INSTRUCTION WORD

EXECUTION TIME

*Incorrect coding - will not generate proper literal.
HOLLERITH LITERAL

EXAMPLE: LDA =3HAAA.DU

ASSEMBLY TIME

INSTRUCTION WORD

EXECUTION TIME
FLOATING-POINT LITERAL

EXAMPLE: FLD = 1.5, DU

ASSEMBLY TIME

INSTRUCTION WORD

EXECUTION TIME
FIXED-POINT LITERAL

EXAMPLE: \( \text{LDA} = 1B5, \text{DU} \)
FIXED-POINT LITERAL

EXAMPLE: $LDX_N = 1B18, DU$

ASSEMBLY TIME

INSTRUCTION WORD

EXECUTION TIME
STORE CHARACTER

**STC (A-Q) WORD, 32**

<table>
<thead>
<tr>
<th>OPERAND ADDR.</th>
<th>STC (A-Q)</th>
<th>TAG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>011010</td>
</tr>
</tbody>
</table>

A OR Q REG.

```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
</table>
```

```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
</table>
```
STORE BYTE

STB(A-Q) WORD,14

OPERAND ADDR. STB(A-Q) 0011XX

A OR Q REG.

0 1 2 3

0 1 2 3
STC1

INSTRUCTION COUNTER

+1

0 17 18 28 29 35

IC + 1 IR 0

INDICATOR REGISTER

ZN CO EE OT PP PM

STC2

INSTRUCTION COUNTER

+2

IC + 2 UNCHANGED
SPECIAL LOAD/STORE

* LBAR
* LDT
LDI
LREG
* MASTER MODE

<table>
<thead>
<tr>
<th>WORD</th>
<th>LREG/SREG</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X0</td>
</tr>
<tr>
<td>1</td>
<td>X2</td>
</tr>
<tr>
<td>2</td>
<td>X4</td>
</tr>
<tr>
<td>3</td>
<td>X6</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
</tr>
<tr>
<td>5</td>
<td>Q</td>
</tr>
<tr>
<td>6</td>
<td>E</td>
</tr>
<tr>
<td>7</td>
<td>TR (SREG only)</td>
</tr>
</tbody>
</table>
SHIFT INSTRUCTIONS

REGISTER

REGISTER LEFT SHIFTS
1. Vacated positions are filled with zeros.
2. Bits vacating bit position zero are lost.

REGISTER

REGISTER RIGHT SHIFTS
1. All vacated positions are filled with the contents of bit zero.
2. Bits vacating bit position 35 are lost.

REGISTER

REGISTER RIGHT LOGICAL SHIFTS
1. All vacated positions are filled with zeros.
2. Bits vacating bit position 35 are lost.

REGISTER

REGISTER LEFT Rotate
The register becomes circular. Bits vacating position zero enter again at bit position 35.
FIXED POINT ARITHMETIC

ADA            SBA
ADQ            SBQ
ADAQ           SBAQ
ADX_n          SBX_n
ASA            SSA
ASQ            SSQ
ASX_n          SSX_n

INDICATORS

ZERO-NEGATIVE-CARRY-OVERFLOW
LOGICAL ADD/SUBTRACT

ADLA        SBLA
ADLQ        SBLQ
ADLAQ       SBLAQ
ADLX_n      SBLX_n

INDICATORS
ZERO-NEGATIVE-CARRY

O’FLO INDICATOR **NOT** AFFECTED
CARRY AND OVERFLOW INDICATORS

* CARRY = SCO
* OVERFLOW = (SCI • SCO) + (SCI • SCO)

EXAMPLES USING ALGEBRAIC ADDITION:

<table>
<thead>
<tr>
<th>4-BIT REPRESENTATION</th>
<th>INDICATOR SET ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. R = +7&lt;br&gt;Y = +1&lt;br&gt;+8</td>
<td>0111&lt;br&gt;0001&lt;br&gt;SCI • SCO</td>
</tr>
<tr>
<td>2. R = -5&lt;br&gt;Y = -4&lt;br&gt;-9</td>
<td>1011&lt;br&gt;1100&lt;br&gt;SCI • SCO</td>
</tr>
<tr>
<td>3. R = -1&lt;br&gt;Y = -4&lt;br&gt;-5</td>
<td>1111&lt;br&gt;1100&lt;br&gt;SCI • SCO</td>
</tr>
<tr>
<td>4. R = -5&lt;br&gt;Y = +4&lt;br&gt;-1</td>
<td>1011&lt;br&gt;0100&lt;br&gt;SCI • SCO</td>
</tr>
</tbody>
</table>
SPECIAL ADD/SUBTRACT

AWCA
AWCQ
AOS
ADL

SWCA
SWCQ

0
A

Q

71

0
SIGN
SMEARED

35
Y
MULTIPLY/DIVIDE

MPY
MULTIPLICAND
MPY-Q  MPF-A

X
MULTIPLIER
MEM. LOC. Y

PRODUCT
COMBINED A-Q REGISTER

DIV
DIVIDEND
Q-REGISTER

÷
DIVISOR
MEM. LOC. Y

REMAINDER
A-REGISTER

QUOTIENT
Q-REGISTER

DVF
DIVIDEND
COMBINED A-Q REGISTER

÷
DIVISOR
MEMORY LOCATION Y

QUOTIENT
A-REGISTER

REMAINDER
Q-REGISTER
TRANSFER CONTROL

UNCONDITIONAL

TRA TSS
RET TSX_n

CONDITIONAL

TZE/TNZ * TOV
TMI/TPL * TEO
TRC/TNC * TEU

* RESETS RESPECTIVE INDICATOR
<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Break</td>
<td>225</td>
</tr>
<tr>
<td>Common Length</td>
<td>0</td>
</tr>
<tr>
<td>V Colat Bits</td>
<td>5</td>
</tr>
<tr>
<td>Primary Symdef Entry</td>
<td></td>
</tr>
<tr>
<td>Float</td>
<td>0</td>
</tr>
<tr>
<td>Secondary Symdef Entry</td>
<td></td>
</tr>
<tr>
<td>Block Length</td>
<td></td>
</tr>
<tr>
<td>Symref</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>ASCII Code</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
</tr>
<tr>
<td>000001</td>
<td>0</td>
</tr>
<tr>
<td>000011</td>
<td>0</td>
</tr>
<tr>
<td>000012</td>
<td>0</td>
</tr>
<tr>
<td>000013</td>
<td>0</td>
</tr>
<tr>
<td>000014</td>
<td>0</td>
</tr>
<tr>
<td>000015</td>
<td>0</td>
</tr>
<tr>
<td>000016</td>
<td>0</td>
</tr>
<tr>
<td>000017</td>
<td>0</td>
</tr>
<tr>
<td>000018</td>
<td>0</td>
</tr>
<tr>
<td>000019</td>
<td>0</td>
</tr>
<tr>
<td>00001A</td>
<td>0</td>
</tr>
<tr>
<td>00001B</td>
<td>0</td>
</tr>
<tr>
<td>00001C</td>
<td>0</td>
</tr>
<tr>
<td>00001D</td>
<td>0</td>
</tr>
<tr>
<td>00001E</td>
<td>0</td>
</tr>
<tr>
<td>00001F</td>
<td>0</td>
</tr>
<tr>
<td>000020</td>
<td>0</td>
</tr>
<tr>
<td>000021</td>
<td>0</td>
</tr>
<tr>
<td>000022</td>
<td>0</td>
</tr>
<tr>
<td>000023</td>
<td>0</td>
</tr>
<tr>
<td>000024</td>
<td>0</td>
</tr>
<tr>
<td>000025</td>
<td>0</td>
</tr>
<tr>
<td>000026</td>
<td>0</td>
</tr>
<tr>
<td>000027</td>
<td>0</td>
</tr>
<tr>
<td>000028</td>
<td>0</td>
</tr>
<tr>
<td>000029</td>
<td>0</td>
</tr>
<tr>
<td>00002A</td>
<td>0</td>
</tr>
<tr>
<td>00002B</td>
<td>0</td>
</tr>
<tr>
<td>00002C</td>
<td>0</td>
</tr>
<tr>
<td>00002D</td>
<td>0</td>
</tr>
<tr>
<td>00002E</td>
<td>0</td>
</tr>
<tr>
<td>00002F</td>
<td>0</td>
</tr>
</tbody>
</table>

Note: The ASCII code column contains the binary representation of the corresponding ASCII characters. The description column lists the operations performed by each instruction.
ERROR LINKAGE

000222 100000000000000 000
000223 204344214320 060

LITERALS

000224 000200000012 000

END

275 is the next available location. GMF VERSION: JFRA/030271, JERB/030271, JHPC/030271.
There were 2 warning flags in the above assembly on page No.
2
<table>
<thead>
<tr>
<th>OCTAL</th>
<th>SYMBOL</th>
<th>REFERENCES BY ALTER NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>GEBDT</td>
<td>24</td>
</tr>
<tr>
<td>222</td>
<td>E.L.</td>
<td>2</td>
</tr>
<tr>
<td>110</td>
<td>MASKA</td>
<td>36 19 36</td>
</tr>
<tr>
<td>140</td>
<td>RCWNR</td>
<td>47 15 18 47</td>
</tr>
<tr>
<td>100</td>
<td>REGTLY</td>
<td>29 4 29</td>
</tr>
<tr>
<td>40</td>
<td>SAVREG</td>
<td>28 28 29 30</td>
</tr>
<tr>
<td>104</td>
<td>SEOBIN</td>
<td>33 14 33</td>
</tr>
<tr>
<td>112</td>
<td>TAIL1</td>
<td>38 21 23 38</td>
</tr>
<tr>
<td>113</td>
<td>TAIL2</td>
<td>39 22 39</td>
</tr>
<tr>
<td>101</td>
<td>TLYREG</td>
<td>30 3 30</td>
</tr>
</tbody>
</table>

**18235 WORDS OF MEMORY WERE USED BY OMAP FOR THIS ASSEMBLY.**

---

<table>
<thead>
<tr>
<th>OCTAL</th>
<th>SYMBOL</th>
<th>REFERENCES BY ALTER NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>46221</td>
<td>01</td>
<td>10-06-71 17,444</td>
</tr>
</tbody>
</table>

UKEFENSE SYMBOLS

MYWNR
SAVING INSTRUCTION COUNTER
IN INDEX REGISTER

MAIN LINE CODING:

RETURN TSX7 SUB STORES IC+1 IN X7

SUBROUTINE CODING:

SUB XXX

TRA 0.7—X7 CONTAINS ADDRESS OF RETURN
SAVING INSTRUCTION COUNTER
IN MEMORY

MAIN LINE CODING:

STC2 BACK STORES IC+2
TRA SUB

RETURN

SUBROUTINE CODING:

SUB XXX

BACK TRA **—INDICATES ADDRESS REPLACED BEFORE USE
SAVING INSTRUCTION COUNTER
AND INDICATORS (STC1)

MAIN LINE CODING:

:  
:  
STC1 BACK STORES IR + IC+1  
TRA SUB  

RETURN  

SUBROUTINE CODING:

SUB XXX  
  
  
EAX7 1  
ASX7 BACK INCREASES ADDRESS PORTION OF  
  
RET BACK BACK BY 1 - ADDRESS OF RETURN  
  
BACK BSS 1 RESTORES INDICATORS AND TRANS-  
  
  
FERS INDIRECT TO RETURN  
  
MEMORY LOCATION FOR INDICATORS  
  
AND IC
SAVING INSTRUCTION COUNTER
AND INDICATORS (STI)

MAIN LINE CODING:

::

STI    BACK    STORES IR
STC2   BACK    STORES IC+2
TRA    SUB

RETURN ☞

SUBROUTINE CODING:

SUB    XXX

::

RET    BACK    RESTORES INDICATORS AND
TRANSFERS INDIRECT

::

BACK    BSS    1    STORAGE FOR IC & IR
SAVING REGISTERS

INDICATORS AND IC

MAIN LINE CODING:

```
.
.
SREG    SAVREG
STI     SAVIND
STC2    SAVIND
TRA     SUB
RETURN
```

SUBROUTINE CODING:

```
SUB       XXX
.
.
LREG    SAVREG
RET     SAVIND
.
.
EIGHT    
SAVREG   BSS  8  JUMP TO 0-MOD-8 LOCATION
SAVIND   BSS  1  STORE REGISTERS HERE
```

SAVE IR & IC
CALL PSEUDO-OPERATION

CODING:

CALL SUB

GENERATED CODING:

TSX1 SUB
TRA * + 2
ZERO .E.L... N*

*N = ALTER NUMBER OF CALL
CALL PSEUDO-OP

EXPANDED VERSION

CODING:

CALL SUB(A1,A2,A3)E1,E2'E,E,I'

GENERATED CODING:

<table>
<thead>
<tr>
<th>TSX1</th>
<th>SUB</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRA</td>
<td>* + 2 + 3 + 2</td>
</tr>
<tr>
<td>ZERO</td>
<td>.E.L...E.I.</td>
</tr>
<tr>
<td>ARG</td>
<td>A1</td>
</tr>
<tr>
<td>ARG</td>
<td>A2</td>
</tr>
<tr>
<td>ARG</td>
<td>A3</td>
</tr>
<tr>
<td>TRA</td>
<td>E2</td>
</tr>
<tr>
<td>TRA</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* NOTE REVERSE ORDER
SAVE PSEUDO OPERATION

EXPANDED VERSION

CODING:

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>SAVE</th>
<th>0.5.7</th>
</tr>
</thead>
</table>

GENERATED INSTRUCTIONS:

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>TRA</th>
<th>**+2+3</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDX0</td>
<td>**.DU</td>
<td></td>
</tr>
<tr>
<td>LDX5</td>
<td>**.DU</td>
<td></td>
</tr>
<tr>
<td>LDX7</td>
<td>**.DU</td>
<td></td>
</tr>
<tr>
<td>RET</td>
<td>.E.L..</td>
<td></td>
</tr>
<tr>
<td>STI</td>
<td>.E.L..</td>
<td></td>
</tr>
<tr>
<td>STX1</td>
<td>.E.L..</td>
<td></td>
</tr>
<tr>
<td>STX0</td>
<td>SYMBOL+1</td>
<td></td>
</tr>
<tr>
<td>STX5</td>
<td>SYMBOL+2</td>
<td></td>
</tr>
<tr>
<td>STX7</td>
<td>SYMBOL+3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CALL SAVE AND RETURN PSEUDO OPS

1
NOW 8 CALL 16 SUB

NOW TSX1 SUB
TRA X+2
ZERO .E.L..,0
XXX

---

SUB SAVE

SUB TRA X+2
RET .E.L..
ST1 .E.L..
STX1 .E.L..

SUBROUTINE BODY

RETURN SUB
TRA SUB+1

NOTE: .E.L.. 

| IC | IR |
CALL SAVE AND RETURN PSEUDO OPS

START   CALL          SUB,(LOC1,LOC2,LOC3),ERR1,ERR2
START   TSX1          SUB       *+7
           TRA          *E.L..,0
           ZERO        LOC1
           ARG         LOC2
           ARG         LOC3
           TRA         ERR2
           TRA         ERR1
           XXX

SUB       SAVE         5,7,2
SUB       TRA          *+5
           LDX5         **,DU
           LDX7         **,DU
           LDX2         **,DU
           RET          .E.L..
           STI          .E.L..
           STX1         .E.L..
           STX5         SUB+1
           STX7         SUB+2
           STX2         SUB+3

*     SUBROUTINE BODY
*     
ERROR1    RETURN       SUB,1
ERROR1    LDX1         .E.L..,*
           SBX1         1,DU
           STX1         .E.L..
           TRA          SUB+1

ERROR2    RETURN       SUB,2
ERROR2    LDX1         .E.L..,*
           SBX1         2,DU
           STX1         .E.L..
           TRA          SUB+1
COMPARE LOGIC FLOW

R \Rightarrow Y

ALGEBRAIC

NO

R \equiv Y

NO

R \equiv Y

NO

R > Y

TMI

NO

R < Y

YES

R = Y

YES

R > Y

LOGICAL

TZE

NO

R \equiv Y

NO

R < Y

TRC

YES

R = Y

YES

R > Y
POSITIVE NUMBERS
AND
COMPARISON INDICATORS

\[
\begin{array}{llll}
\text{REGISTER} &=& 8 & 001000 \\
\text{MEMORY} &=& 8 & 001000 \\
\end{array}
\]

\[
\begin{array}{llll}
001000 & 001000 & 111000 & 1000000
\end{array}
\]

\[
\begin{array}{llll}
\text{ZERO} & \\
\text{SCO} & \text{SCI} & \text{CARRY}
\end{array}
\]

\[
\begin{array}{llll}
\text{REGISTER} &=& 8 & 001000 \\
\text{MEMORY} &=& 7 & 000111 \\
\end{array}
\]

\[
\begin{array}{llll}
001000 & 001000 & 111001 & 1000001
\end{array}
\]

\[
\begin{array}{llll}
\text{SCO} & \text{SCI} & \text{CARRY}
\end{array}
\]

\[
\begin{array}{llll}
\text{REGISTER} &=& 8 & 001000 \\
\text{MEMORY} &=& 9 & 001001 \\
\end{array}
\]

\[
\begin{array}{llll}
001000 & 001000 & 110111 & 1111111
\end{array}
\]

NEGATIVE
NEGATIVE NUMBERS AND COMPARISON INDICATORS

REGISTER = -8
MEMORY = -8

\[
\begin{array}{c}
\text{REGISTER} = -8 \\
\text{MEMORY} = -8 \\
\end{array}
\]

\[
\begin{array}{c}
\text{REGISTER} = -8 \\
\text{MEMORY} = -7 \\
\end{array}
\]

\[
\begin{array}{c}
\text{REGISTER} = -8 \\
\text{MEMORY} = -9 \\
\end{array}
\]

\[
\begin{array}{c}
\text{REGISTER} = -8 \\
\text{MEMORY} = -7 \\
\end{array}
\]

\[
\begin{array}{c}
\text{REGISTER} = -8 \\
\text{MEMORY} = -9 \\
\end{array}
\]

ZERO
CARRY
NEGATIVE
CARRY
COMPARISON INDICATORS

POSITIVE REGISTER - NEGATIVE MEMORY

\[
\begin{array}{ccc}
\text{REGISTER} & = & 8 \\
\text{MEMORY} & = & -8 \\
001000 & & 001000 \\
111000 & & 010000 \\
\end{array}
\]

NO INDICATORS SET

\[
\begin{array}{ccc}
\text{REGISTER} & = & 8 \\
\text{MEMORY} & = & -7 \\
001000 & & 001000 \\
111001 & & 001111 \\
\end{array}
\]

NO INDICATORS SET

NEGATIVE REGISTER - POSITIVE MEMORY

\[
\begin{array}{ccc}
\text{REGISTER} & = & -8 \\
\text{MEMORY} & = & 8 \\
111000 & & 111000 \\
001000 & & 1110000 \\
\end{array}
\]

SCC - SCC
NEGATIVE CARRY
### Algebraic Comparison

<table>
<thead>
<tr>
<th>ZERO</th>
<th>NEGATIVE</th>
<th>CARRY</th>
<th>RELATION</th>
<th>SIGN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$C(A) &gt; C(Y)$</td>
<td>$C(A)_0 = 0$, $C(Y)_0 = 1$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$C(A) &gt; C(Y)$</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$C(A) = C(Y)$</td>
<td>$C(A)_0 = C(Y)_0$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$C(A) &lt; C(Y)$</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>$C(A) &lt; C(Y)$</td>
<td>$C(A)_0 = 1$, $C(Y)_0 = 0$</td>
</tr>
</tbody>
</table>

### Logical Comparison

<table>
<thead>
<tr>
<th>ZERO</th>
<th>CARRY</th>
<th>RELATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$C(A) &lt; C(Y)$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$C(A) = C(Y)$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$C(A) &gt; C(Y)$</td>
</tr>
</tbody>
</table>
FIXED POINT COMPARE

CMPA    Y
CMPQ    Y
CMPAQ   Y-PAIR
CMPX<sub>N</sub>  Y<sub>0...17</sub>
SPECIAL PURPOSE COMPARES

ALGEBRAIC COMPARE WITH LIMITS

CWL Y

COMPARE MAGNITUDE

CMG Y

COMPARE MASKED

CMK Y
### Boolean Operations

#### Truth Table

<table>
<thead>
<tr>
<th>ANDING</th>
<th>ELIMINATES BITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANA</td>
<td>ANSA</td>
</tr>
<tr>
<td>ANQ</td>
<td>ANSQ</td>
</tr>
<tr>
<td>ANAQ</td>
<td>ANCXN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ORING</th>
<th>INSERTS BITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORA</td>
<td>ORSA</td>
</tr>
<tr>
<td>ORQ</td>
<td>ORSQ</td>
</tr>
<tr>
<td>ORAQ</td>
<td>ORSXN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EXCLUSIVE OR</th>
<th>BINARY ADD (NO CARRY)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERA</td>
<td>ERSA</td>
</tr>
<tr>
<td>EQ</td>
<td>ERSQ</td>
</tr>
<tr>
<td>ERAQ</td>
<td>ERSXN</td>
</tr>
</tbody>
</table>

91
EXAMPLE OF ANDING

ANA  MASK
ANA  = 020401020402

AREG BEFORE  0 7 0 6 0 5 0 4 0 3 0 2
AREG BEFORE 000 111 000 110 000 101 000 100 000 011 000 010
MASK 000 010 000 100 000 001 000 010 000 100 000 010
AREG AFTER 000 010 000 100 000 001 000 000 000 000 010
AREG AFTER 0 2 0 4 0 1 0 0 0 0 0 2
# Example of ORing

<table>
<thead>
<tr>
<th>ORA</th>
<th>MASK</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORA = 020401020402</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AREG BEFORE</th>
<th>0 7 0 6 0 5 0 4 0 3 0 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>AREG BEFORE</td>
<td>000 111 000 110 000 101 000 100 000 011 000 010</td>
</tr>
<tr>
<td>MASK</td>
<td>000 010 000 100 000 001 000 010 000 100 000 010</td>
</tr>
<tr>
<td>AREG AFTER</td>
<td>000 111 000 110 000 101 000 110 000 111 000 010</td>
</tr>
<tr>
<td>AREG AFTER</td>
<td>0 7 0 6 0 5 0 6 0 7 0 2</td>
</tr>
</tbody>
</table>
**EXAMPLE OF THE EXCLUSIVE OR**

<table>
<thead>
<tr>
<th>ERA</th>
<th>MASK</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERA</td>
<td>=020401020402</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AREG BEFORE</th>
<th>0</th>
<th>7</th>
<th>0</th>
<th>5</th>
<th>0</th>
<th>5</th>
<th>0</th>
<th>4</th>
<th>0</th>
<th>3</th>
<th>0</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>AREG BEFORE</td>
<td>000</td>
<td>111</td>
<td>000</td>
<td>110</td>
<td>000</td>
<td>101</td>
<td>000</td>
<td>100</td>
<td>000</td>
<td>011</td>
<td>000</td>
<td>010</td>
</tr>
<tr>
<td>MASK</td>
<td>000</td>
<td>010</td>
<td>000</td>
<td>100</td>
<td>000</td>
<td>001</td>
<td>000</td>
<td>010</td>
<td>000</td>
<td>100</td>
<td>000</td>
<td>010</td>
</tr>
<tr>
<td>AREG AFTER</td>
<td>000</td>
<td>101</td>
<td>000</td>
<td>010</td>
<td>000</td>
<td>100</td>
<td>000</td>
<td>110</td>
<td>000</td>
<td>111</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>AREG AFTER</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
BOOLEAN COMPARE OPERATIONS

COMPARATIVE AND

CANA  Y
CANQ  Y
CANAQ  Y-PAIR
CANX_N  Y

COMPARATIVE NOT AND

CNAAN  Y
CNAQ  Y
CNAAQQ  Y-PAIR
CNAX_N  Y
COMPARATIVE AND

CANA
TNZ

MASK
BITON

MASK  0010
AREG  1111
Z=  0010

CANA  COMPARATIVE AND TO AREG
CANQ  COMPARATIVE AND TO QREG
CANAQ  COMPARATIVE AND TO AQREG
CANXN  COMPARATIVE AND TO XREG
COMPARATIVE NOT AND

CNA A  MASK
TNZ  BITON

MASK  1101
1ST COMPLEMENT  0010
AREG  1111
Z=  0010

CNA A  COMPARATIVE NOT AND TO AREG
CNAQ  COMPARATIVE NOT AND TO QREG
CNA AQ  COMPARATIVE NOT AND TO AQREG
CNA X N  COMPARATIVE NOT AND TO XREG
EXAMPLE OF
BOOLEAN OPERATIONS
AND
BOOLEAN COMPARES

CHECK   LDA   SWITCH
        ANA   =Ø241000,DU
        CANA  =Ø200000,DU
        TNZ   BCD
        CANA  =Ø040000,DU
        TNZ   ASCII
        CNA   =Ø776000,DU
        TNZ   BINARY

ERROR   LDA   =Ø400000,DU
        ORSA  SWITCH
        TRA   COMRET

BCD     CANA  =Ø041000,DU
        TNZ   ERROR

OKBCD   LDA   =Ø200040,DU
        ERS   SWITCH
        TRA   COMRET

NOKBCD  LDA   =Ø200004,DU
        ERS   SWITCH
        COMRET TRA   $X$
INDIRECT THEN TALLY (IT) MODIFICATION

VARIATIONS

I   INDIRECT
ID  INCREMENT ADDRESS - DECREMENT TALLY
DI  DECREMENT ADDRESS - INCREMENT TALLY
AD  ADD DELTA
SD  SUBTRACT DELTA
*CI CHARACTER FROM INDIRECT
*SC SEQUENCE CHARACTER
*SCR SEQUENCE CHARACTER REVERSE
F   FAULT
IDC INCREMENT ADDRESS - DECREMENT TALLY
     AND CONTINUE
DIC DECREMENT ADDRESS - INCREMENT TALLY
     AND CONTINUE

*CHARACTER HANDLING
INDIRECT -I-
EXAMPLE

1 8 1
TSX1 SUBR

SUBR STX1 ERLKG

TRA ERLKG, I

ERLKG BSS 1
1. Fetch TALLY indirect word.
2. Use operand address for instruction execution.
3. TALLY indirect word is not altered in memory.
INDIRECT VARIATION (I)

FLOWCHART

START

FETCH THE INDIRECT WORD

ADDRESS FIELD IS EFF ADDRESS

EXECUTE THE OPERATION

EXIT
INDIRECT WORD Formats for IT Modification

51 I
Address
0 17 18

56 ID
54 DI
Address Tally
0 17 18 29 30 35

52 SC
5 SCR
Address Tally z Char. Pos.
0 17 18 29 30 32 33 35

50 CI
Address
0 17 18

53 AD
44 SD
Address Tally Delta
0 17 18 29 30 32 35

57 IDC
55 DIC
Address Tally
0 17 18 29 30 31 32
ADDRESS TALLY PSEUDO-OPERATIONS

TALLY A, T, C, (TALLY)
used for I, ID, DI, SC and CI types of tally modification, where SC and CI are for six (6) bit characters.

TALLY B A, T, B, (TALLY BYTE)
used for SC and CI types of tally modification, where nine (9) bit bytes (characters) are desired.

TALLY D A, T, D, (TALLY AND DELTA)
used for AD and SD types of tally modification.

TALLY C A, T, MOD, (TALLY AND CONTINUE)
used for IDC and DIC types of tally modification.

ABBREVIATIONS:

<table>
<thead>
<tr>
<th>CODE</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ADDRESS</td>
</tr>
<tr>
<td>T</td>
<td>TALLY</td>
</tr>
<tr>
<td>C</td>
<td>CHARACTER POSITION</td>
</tr>
<tr>
<td>B</td>
<td>BYTE</td>
</tr>
<tr>
<td>D</td>
<td>DELTA (Δ)</td>
</tr>
<tr>
<td>MOD</td>
<td>MODIFIER</td>
</tr>
</tbody>
</table>
1. Fetch tally indirect word
2. Use operand address for instruction execution.
3. Add one (1) to operand address and restore in tally word.
4. Subtract one (1) from tally and restore in tally word.
5. Restore updated tally indirect word to memory.
INCREMENT ADDRESS-DECREMENT TALLY (ID)

FLOWCHART

START

FETCH THE TALLY WORD

ADDRESS FIELD IS EFF ADDRESS

EXECUTE THE OPERATION

ADD ONE TO ADDRESS FIELD

SUBTRACT ONE FROM TALLY

RESTORE TALLY WORD TO MEMORY

EXIT
INDEXING EXAMPLE

PROBLEM: MOVE 50 WORDS OF DATA FROM HERE TO THERE.

1 8 6
LDX7 50,DU
LDX6 0,DU
GO
LDA HÈRE,6
STA TÈRÈ,6
ADX6 1,DU
SBX7 1,DU
TNZ GO

INDIRECT THEN TALLY -ID- EXAMPLE

LDA TAL1,ID
STA TAL2,ID
TTF ×-2

TAL1 TALLY HÈRE,50
TAL2 TALLY TÈRÈ,50
I AND ID EXAMPLE

PROBLEM: SEARCH A 100 WORD TABLE AND COMPLEMENT ALL NEGATIVES FOUND THEREIN.

1 8 1 6

START LDA TAL1, I
TPL 1
NEG +2
STA TAL1, ID
TTF START

... TAL1 TALLY TABLE, 100
TALLY REFRESHING

1 8 6

LDA TALREF
STA TAL1

TALREF TALLY TABLE, 100
TAL1 BSS 1
1. FETCH TALLY INDIRECT WORD
2. SUBTRACT ONE (1) FROM OPERAND ADDRESS AND USE.
3. RESTORE UPDATED ADDRESS IN TALLY WORD.
4. ADD ONE (1) TO TALLY AND RESTORE IN TALLY WORD.
5. RESTORE UPDATED TALLY INDIRECT WORD TO MEMORY.
DECREMENT ADDRESS-INCREMENT TALLY (DI)

FLOWCHART

START

FETCH THE TALLY WORD

SUBTRACT ONE FROM THE ADDR. FIELD

ADD ONE TO THE TALLY FIELD

ADDR FIELD IS NOW THE EFF ADDR

EXECUTE THE OPERATION

RESTORE TALLY WORD TO MEMORY

EXIT
DECREMENT ADDRESS INCREMENT TALLY -DI-
EXAMPLE

PROBLEM: TRANSFER 500 WORDS OF DATA FROM ONE
TABLE TO ANOTHER IN A REVERSE ORDER.

1 8 6

LDAQ TALR
STAQ TAL1
LDA TAL1,1D
STA TAL2,DI
TTF ×2

TALR ETALLY FORWRD,500
TALLY REVRSE,-500
TAL1 BSS 1
TAL2 BSS 1
FORWRD BSS 500
REVRSE BFS 500
I, ID AND DI EXAMPLE

PROBLEM: REVERSE THE ORDER OF THE WORDS OF A 500 WORD TABLE WITHIN ITSELF.

1  8  6

GOOD LDA TAL1,1
LDQ TAL2,DI
STA TAL2,1
STQ TAL1,ID
TTF GOOD

TAL1 TALLY TABLE,250
TAL2 TALLY TABLE+500,-250
TABLE BSS 500
1. Fetch tally indirect word.
2. Use operand address for instruction execution.
3. Add delta to operand address and restore in tally word.
4. Subtract one (1) from tally and restore in tally word.
5. Restore updated tally indirect word to memory.
ADD DELTA VARIATION (AD)

FLOWCHART

START

FETCH THE TALLY WORD

ADDRESS FIELD IS EFF ADDR

EXECUTE THE OPERATION

ADD DELTA TO THE ADDR FIELD

SUBTRACT ONE FROM TALLY

RESTORE TALLY WORD TO MEMORY

EXIT
1. Fetch Tally Indirect Word
2. Subtract Delta from Operand Address and Use.
3. Restore Updated Address in Tally Word.
4. Add One (1) to Tally and Restore in Tally Word.
5. Restore Updated Tally Indirect Word to Memory.
SUBTRACT DELTA VARIATION (SD)

FLOWCHART

START

FETCH THE TALLY WORD

SUBTRACT DELTA FROM ADDR FIELD

ADD ONE TO THE TALLY

ADDR FIELD IS NOW THE EFF ADDR

EXECUTE THE OPERATION

RESTORE TALLY WORD TO MEMORY

EXIT
ADD DELTA -AD- AND SUBTRACT DELTA -SD-

**Problem:** Reverse the order of one table into another using double precision commands.

```
1  8  6

BEGIN  LDÂQ  TAL1, AD
       STÂQ  TAL2, SD
       TTF  BEGIN

TAL1  TÂLLYD  TABLEA, 4096, 2
TAL2  TÂLLYD  TABLEB, 2
TABLEAEBSS  8192
TABLEB BFS  8192
```
CHARACTER OPERATIONS

OPERAND FROM MEMORY
SIX-BIT CHARACTER

OPERAND FROM MEMORY

0 1 2 3 4 5
X X X CHAR X X

TO PROCESSOR REGISTER A OR Q

0 0 0 0 0 CHAR
0 1 2 3 4 5
CHARACTER OPERATIONS

OPERAND FROM MEMORY

NINE-BIT CHARACTER

OPERAND FROM MEMORY

\[
\begin{array}{cccc}
X & X & \text{CHAR} & X \\
0 & 1 & 2 & 3 \\
\end{array}
\]

TO PROCESSOR REGISTER A OR Q

\[
\begin{array}{cccc}
0 & 0 & 0 & \text{CHAR} \\
0 & 1 & 2 & 3 \\
\end{array}
\]
CHARACTER OPERATIONS

SIX-BIT CHARACTER

FROM A OR Q

<table>
<thead>
<tr>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>CHAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

TO MEMORY

<table>
<thead>
<tr>
<th>X</th>
<th>X</th>
<th>X</th>
<th>CHAR</th>
<th>X</th>
<th>X</th>
<th>X</th>
</tr>
</thead>
</table>
NOTE:

CI TALLY INDIRECT WORD NOT ALTERED IN MEMORY.
CHARACTER FROM INDIRECT VARIATION (CI)

FLOWCHART

START

FETCH THE INDIRECT WORD

ADDR FIELD IS EFF ADDR

EXECUTE THE OPERATION ON CHARACTER SPECIFIED BY COUNT FIELD

EXIT
SEQUENCE CHARACTER VARIATION (SC)

FLOWCHART

START

FETCH TALLY AND USE ADDR AS EFF ADDR

EXECUTE THE OPERATION

SUBTRACT ONE FROM THE TALLY

ADD ONE TO THE CHARACTER COUNT POSITION

CC > 5?

Y

ADD ONE TO THE TALLY WORD ADDRESS

N

RESTORE TALLY WORD TO MEMORY

EXIT

SET CHAR POSITION TO ZERO

123
SEQUENCE CHARACTER REVERSE (SCR)
FLOWCHART

START

FETCH THE TALLY WORD

SUBTRACT ONE FROM CHARACTER COUNT

IS COUNT NEGATIVE?

YES

SET CHARACTER COUNT TO 5

SUBTRACT ONE FROM ADDRESS

ADD ONE TO TALLY

EXECUTE OPERATION

RESTORE TALLY WORD TO MEMORY

EXIT

123.1
REVERSING TABLE WITHIN ITSELF

CHARACTER BY CHARACTER

LDA    TAL1,SCR
LDQ    TAL2,CI
STQ    TAL1,CI
STA    TAL2,SC
TTF    *-4

TAL1    TALLY    TABLE+200,0,0
TAL2    TALLY    TABLE+500,0
TABLE   BSS      200
-Ci- AND -Sc-

PROBLEM: GIVEN

DOUBLE-WORD XXXXXX XX.XXX

INSERT *'s IN PLACE OF LEADING ZEROS

1 8 6

GO LDQ =3H06* DL
LDA TALI, CI
TNZ *+3
STQ' TALI, SC
TRA GO+1

TALI TALLY WORD,,0
PROBLEM:

GIVEN
DOUBLE-WORD xxxxxxx xxxxxxx
INSERT DEC PT AT IMPLIED LOC

LDQ =3H,DL
LDA TAL1,SC
STA TAL2,SC
TTF *-2
STQ TAL2,CI

TAL1 TALLY WORD,8,1
TAL2 TALLY WORD,8,0
CHECK PROTECTION AND DECIMAL INSERTION

EXAMPLES:
BEFORE     0000XX XXXXXX     AFTER     ***XXX XX.XXX
BEFORE     000000 000XXX     AFTER     ******* **.XXX

8          16
LDQ =3H8.*\,DL
LDA TAL1,SC
TNZ CHAR
STQ TAL2,SC
TTF -3,1C
TRA INSERT
LDA TAL1,SC
CHAR STA TAL2,SC
TTF *-2
INSERT QRS 6
STQ TAL2,CI

TAL1 TALLY WORD,8,1
TAL2 TALLY WORD,8,0
1. Fetch TALLY INDIRECT WORD.
2. Trap operand address and modification for instruction execution.
3. Add one (1) to operand address and restore in TALLY word.
4. Subtract one (1) from TALLY and restore in TALLY word.
5. Restore updated TALLY INDIRECT word to memory.
6. Use trapped address for further Indirection if indicated by the tag field.
1. FETCH TALLY INDIRECT WORD
2. SUBTRACT ONE (1) AND TRAP OPERAND ADDRESS AND MODIFICATION.
3. RESTORE UPDATED OPERAND ADDRESS IN TALLY WORD.
4. ADD ONE (1) TO TALLY AND RESTORE IN TALLY WORD.
5. RESTORE UPDATED TALLY INDIRECT WORD TO MEMORY.
6. USE TRAPPED ADDRESS FOR FURTHER INDIRECTION IF INDICATED BY THE TAG FIELD.
REPEAT INSTRUCTION

INSTRUCTION FORMAT

<table>
<thead>
<tr>
<th>TALLY</th>
<th>EXIT CONDITIONS</th>
<th>OP CODE</th>
<th>I</th>
<th>DELTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-256</td>
<td></td>
<td></td>
<td>1</td>
<td>0-63</td>
</tr>
</tbody>
</table>

CODING FORMAT

RPT

1 8 16

TALLY, DELTA, EXIT CONDITIONS

SUMMARY

1. REPEAT NEXT INSTRUCTION N TIMES.
2. INCREMENT OPERAND ADDRESS BY DELTA.
3. EXIT PRIOR TO TALLY RUNOUT IF EXIT CONDITIONS MET.
REPEAT INSTRUCTION FORMAT

- **TALLY** (1-256)
- **EXIT CONDITIONS**
- **OP CODE RPT** (520)
- **DELTA** (1-63)

**Inhibit Interrupts**

- **ZERO ON**
- **ZERO OFF**
- **NEG ON**
- **NEG OFF**
- **CARRY ON**
- **CARRY OFF**
- **OFLO ON**
- **TZE**
- **TNZ**
- **TMI**
- **TPL**
- **TRC**
- **TNC**
- **TOV**
REPEAT INSTRUCTION

RULES:

REPEATED INSTRUCTION MUST BE MODIFIED BY INDEX REGISTER.

ONLY X1 THRU X7 MAY BE USED.

TALLY, EXIT CONDITIONS AUTOMATICALLY PLACED IN XO.

TALLY DECREMENTED IN XO.

CODING EXAMPLE:

<table>
<thead>
<tr>
<th>1</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>KEY</td>
<td></td>
</tr>
<tr>
<td>EAX4</td>
<td>TABLE</td>
<td></td>
</tr>
<tr>
<td>RPT</td>
<td>64,1,TZE</td>
<td></td>
</tr>
<tr>
<td>CMPA</td>
<td>0,4</td>
<td></td>
</tr>
<tr>
<td>TZE</td>
<td>FOUND</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TABLE</td>
<td>BSS</td>
<td>64</td>
</tr>
<tr>
<td>KEY</td>
<td>BSS</td>
<td>1</td>
</tr>
</tbody>
</table>
ADDRESS MODIFICATION
OF
REPEATED INSTRUCTION

FIRST TIME:
EFFECTIVE ADDRESS = TA + C(x_n)

EA + \triangle \Rightarrow C(x_n)

N TIMES:
EFFECTIVE ADDRESS = C(x_n)

EA + \triangle \Rightarrow C(x_n)

TA = Tentative Address
EA = Effective Address
CODING EXAMPLE
USING
REPEAT INSTRUCTION

PROBLEM: SEARCH A TABLE OF 64 WORDS FOR NEGATIVE VALUES. STORE ZEROS IN EVERY NEGATIVE WORD.

GO        EAX7        TABLE
RPT       64,1,TMI
LDA       0.7
TPL       DONE
STZ       -1.7
TRA       GO
DONE      XXX
          .
          .
TABLE     BSS        64
RPTX - RPDX

0 10 1718 35

OP CODE 1 DELTA

PROGRAMMER LOADS XO:

0 17

TALLY A B C EXIT CONDITIONS

RPTX DELTA RPDX ,DELTA
CODING EXAMPLES

GO
EAX7
RPT 64,1,TMI
LDA 0,7
TPL DONE
STZ -1,7
TRA GO

EAX7 TABLE
RPT 64,1,TMI
LDA ,7
TPL DONE
STZ -1,7
CANXO =Ø776000,DU
TZE DONE
RPTX ,1
LDA ,7
TMI BACK
DONE .
.
EAX7 TABLE
LDX0 GO
GO
RPTX 64,1,TMI
LDA 0,7
TPL DONE
STZ -1,7
CANXO =Ø776000,DU
TNZ GO
### Repeat Double Format

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>8</th>
<th>9</th>
<th>11</th>
<th>17</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>TALLY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A B C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONDITIONS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OP CODE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I DELTA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0-63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**RPD**  
**RPDA**  
**RPDB**

**RPDA**  
ADD DELTA ONLY TO FIRST REPEATED INSTRUCTION

**RPDB**  
ADD DELTA ONLY TO SECOND REPEATED INSTRUCTION
REPEAT DOUBLE EXAMPLES

EAX6 FROM
EAX7 TO
RPD 100,2
LDAQ 0,6
STAQ 0,7

FROM BSS 200
TO BSS 200

SAME INDEX REGISTER MAY NOT BE USED FOR BOTH INSTRUCTIONS

LDI ,DL
LDA CARDIN
EAX2 CARDIN+2
EAX3 0
RPDA 22,1
ADLA 0,2
AWCA =0,3
CMPA CARDIN+1
TNZ ERROR
REPEAT LINK (RPL)

INSTRUCTION FORMAT:

<table>
<thead>
<tr>
<th>TALLY</th>
<th>EXIT CONDITIONS</th>
<th>OP CODE</th>
</tr>
</thead>
</table>

CODING FORMAT:

RPL    TALLY,EXIT CONDITIONS

ADDRESS MODIFICATION:

FIRST TIME

EFFECTIVE ADDRESS = TA + C(Xn)

EA ⇒ C(Xn)

N TIMES

NEW EFFECTIVE ADDRESS = C(EAO...17)

C(NEAO...17) ⇒ C(Xn)
RPL CODING

EXAMPLE

EAX7   A
LCQ    = 1,DU
LDA    = 3H1DD,DL
RPL    5,TZE
CMK    0,7
TNZ    ERROR

.:.
.:.

139.1
EXECUTION OF RPL

A - REG

1st TIME

B

C

D

E

2nd TIME

3rd TIME

4th TIME
BCD INSTRUCTION FLOWCHART

LDA = 017
EAQ = 0
BCD = 80
BCD = 64

START

MULTIPLY A REG BY 8 BY SHIFTING LEFT THREE BITS

DIV A-REG BY CONST SAVING 4 BIT QUOT & RMDR, THEN SHIFT Q-REG 6 BITS LEFT SENDING QUOT Q-REG (32-35)

MULTIPLY A REG BY 8 BY SHIFTING LEFT THREE BITS

DIV A-REG BY CONST SAVING 4 BIT QUOT & RMDR, THEN SHIFT Q-REG 6 BITS LEFT SENDING QUOT Q-REG (32-35)

EXIT

A REGISTER
000000000017
000000000170
000000000050
0000000000500
00000000000000

Q REGISTER
000000000000
000000000000
00000000000001
00000000000001
000000000000105
## Binary to BCD Conversion Constants

<table>
<thead>
<tr>
<th>Conversion Step</th>
<th>(-10^{10}+1)</th>
<th>(-10^9+1)</th>
<th>(-10^8+1)</th>
<th>(-10^7+1)</th>
<th>(-10^6+1)</th>
<th>(-10^5+1)</th>
<th>(-10^4+1)</th>
<th>(-10^3+1)</th>
<th>(-10^2+1)</th>
<th>(-10^1+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8^{1} \times 10^{9}</td>
<td>8 \times 10^{8}</td>
<td>8 \times 10^{7}</td>
<td>8 \times 10^{6}</td>
<td>8 \times 10^{5}</td>
<td>8 \times 10^{4}</td>
<td>8 \times 10^{3}</td>
<td>8 \times 10^{2}</td>
<td>8 \times 10^{1}</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>8^{2} \times 10^{8}</td>
<td>8^{2} \times 10^{7}</td>
<td>8^{2} \times 10^{6}</td>
<td>8^{2} \times 10^{5}</td>
<td>8^{2} \times 10^{4}</td>
<td>8^{2} \times 10^{3}</td>
<td>8^{2} \times 10^{2}</td>
<td>8^{2} \times 10^{1}</td>
<td>8^{2}</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8^{3} \times 10^{7}</td>
<td>8^{3} \times 10^{6}</td>
<td>8^{3} \times 10^{5}</td>
<td>8^{3} \times 10^{4}</td>
<td>8^{3} \times 10^{3}</td>
<td>8^{3} \times 10^{2}</td>
<td>8^{3} \times 10^{1}</td>
<td>8^{3}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8^{4} \times 10^{6}</td>
<td>8^{4} \times 10^{5}</td>
<td>8^{4} \times 10^{4}</td>
<td>8^{4} \times 10^{3}</td>
<td>8^{4} \times 10^{2}</td>
<td>8^{4} \times 10^{1}</td>
<td>8^{4}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>8^{5} \times 10^{5}</td>
<td>8^{5} \times 10^{4}</td>
<td>8^{5} \times 10^{3}</td>
<td>8^{5} \times 10^{2}</td>
<td>8^{5} \times 10^{1}</td>
<td>8^{5}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>8^{6} \times 10^{4}</td>
<td>8^{6} \times 10^{3}</td>
<td>8^{6} \times 10^{2}</td>
<td>8^{6} \times 10^{1}</td>
<td>8^{6}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8^{7} \times 10^{3}</td>
<td>8^{7} \times 10^{2}</td>
<td>8^{7} \times 10^{1}</td>
<td>8^{7}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8^{8} \times 10^{2}</td>
<td>8^{8} \times 10^{1}</td>
<td>8^{8}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8^{9} \times 10^{1}</td>
<td>8^{9}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>8^{10}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
BCD INSTRUCTION

CONVERT 1460 OCTAL TO 816 DECIMAL

A-REG

00 00 00 00 14 60 = 816

QLS 6

00 00 00 00 00 00

ALS 3

00 00 00 01 46 00 = 6528

00 00 00 00 00 10

= 8 R128

00 00 00 00 02 00 = 128

QLS 6

00 00 00 00 00 10

ALS 3

00 00 00 00 20 00 = 1024

00 00 00 00 10 01

= 1 R384

00 00 00 00 06 00 = 384

QLS 6

00 00 00 10 01 00

ALS 3

00 00 00 00 60 00 = 3072

00 00 00 10 01 06

= 6 R0

00 00 00 00 00 00

= 8 1 6

00 00 00 10 01 00
BCD INSTRUCTION

CONVERTING VALUES TO 999999

EAX2 0 PLACE ZEROS IN X2
LDA X LOAD ACCUMULATOR WITH VALUE TO BE CONVERTED
RPT 6,1 REPEAT 6 TIMES, INCREMENT BY 1
BCD TAB.2 DIVIDE BY TAB, TAB+1, ETC.
STQ Y STORE CONVERTED NUMBER IN Y

TAB DEC 800000, 640000, 512000, 409600, 327680
DEC 262144
BCD EXAMPLE
CONVERTING MAXIMUM VALUES

EAX7  TABLE  PLACE ADDRESS OF TABLE IN X7
EAQ   0      ZERO OUT QR
LDA   VALUE  BINARY VALUE TO BE CONVERTED
RPT   4.1    CONVERT FIRST 4 DIGITS
BCD   0.7    STEP THRU TABLE
STQ   RESULT STORE Q IN MEMORY

* MEMORY NOW CONTAINS 00XXXX
RPT   6.1    CONVERT REMAINING VALUE
BCD   0.7    X7 CONTAINS ADDRESS OF TABLE+4
STQ   RESULT+1

TABLE
DEC   8E9B35,64E8B35,512E7B35
DEC   4096E6B35,3276800000
DEC   2621440000,2097152000
DEC   1677721600,1342177280
DEC   1073741824
XED CODING EXAMPLE

EXAMPLE: A PROGRAM IS MADE UP OF FOUR (4) DIVISIONS WHICH MAY BE ACCESSED AT DIFFERENT TIMES.

X7 INITIALIZED TO DETERMINE ENTRY.

XED LOC,7

RE-ENTRY → XXX

LOC ESTC1 SAVE
TRA FIRST
STC1 SAVE
TRA SECOND
STC1 SAVE
TRA THIRD
STC1 SAVE
TRA FOURTH

FIRST →
TRA SAVE,I

SECOND →
TRA SAVE,I

THIRD →
TRA SAVE,I

FOURTH →
TRA SAVE,I
EXAMPLES OF PSUEDO OPERATIONS

...ALSO LITERALS AND MACROS.

PREFACE
PROGRAM BREAK 2206
COMMON LENGTH 24
V COUNT BITS 5

PRIMARY SYMDEF ENTRY
.SMDB 2001
.SAVE 2130

SECONDARY SYMDEF ENTRY
.EVN 2012
.BLOCK LENGTH

1 LABCOM 20
2 TALLYS 4

SYMREF
3 .SMREF
EXAMPLES OF PSUEDO OPERATIONS

1. TTL
2. TILS  ALSO LITERALS AND MACROS.
3. -------------------------------
4. **
5. ** THE FOLLOWING PROGRAM LISTING
6. ** WILL SHOW MOST OF THE COMMONLY
7. ** USED PSUEDO OPS IN A
8. ** RELOCATABLE ASSEMBLY.
9. **
10. **
11. **
12. **
13. **
14. **
15. **
16. **
17. ** THIS FIRST PSUEDO OP IS NECESSARY FOR ALL RELOCATABLE PROGRAMS IN ORDER
18. ** TO PROVIDE AN ENTRY POINT FOR GLOAD. IF MORE THAN ONE SYMBOL APPEARS IN
19. ** THE VARIABLE FIELD THE FIRST SYMBOL WILL BE TAKEN AS THE ENTRY POINT.
20. **
21. SYMDEF .SMODEF PROVIDES AN ENTRY POINT INTO THE PROGRAM.
22. SYMDEF --EVEN A MINUS SIGN DENOTES A SECONDARY SYMDEF.
23. **
24. ** THE PCC WILL ALLOW THE PRINTING OF ALL PSUEDO OPS FOR YOUR CONVENIENCE.
25. **
26. PCC ON
27. **
28. ** THE ORG PSUEDO OP WILL START THE PROGRAM OFF AT LOCATION 2000 OCTAL.
29. **
30. 002000 ORG 1004
31. 002001 .ORG, LDA .ORG,
32. 002002 .SMODEF STA .OPSIN
33. START LDX7 .ZERO.
34. ** THE EQU PSUEDO OP WILL EQUATE THE SYMBOLIC ADDRESS OF .EQU WITH 512 DECIMAL.
35. **
36. 001000 .EQU, EQU 512
37. 001003 LDA .EQU.
38. **
39. ** IN THIS EXAMPLE OF THE EQU PSUEDO OP, WE ARE EQUATING ONE SYMBOL....
40. ** TO A PREVIOUSLY DEFINED SYMBOL. HOWEVER, THIS SYMBOL MUST NOT BE A......
41. ** SYMDEF OR SYMREF.
42. **
43. 002002 STOP EQU START
44. 002004 LDA START
45. 002005 LDA STOP
46. **
47. ** THE BOOL PSUEDO OP WILL EQUATE .BOOL. WITH 1200 OCTAL.
48. **
49. 001200 .BOOL BOOL 1200
50. 002006 STA .BOOL+6
EXAMPLES OF PSEUDO OPERATIONS

... ALSO LITERALS AND MACROS.

51 * THE NULL PSEUDO OP CAUSES THE LABEL TO ASSUME THE NEXT LOCATION COUNTER VALUE.
52 * .NULL. NULL
53 .NULL. NULL
54 LDA .ARG.
55 LDA .E.L.
56 .E.L. IS THE ERROR LINKAGE SYMBOL.
57 * THIS EVEN PSEUDO OP FORCES THE LOCATION COUNTER TO AN EVEN COUNT
58 *
59 * EVEN
60 .EVN LDA .VFD.
61 STA .ODD
62 *
63 * THE ODD PSEUDO OP FORCES THE LOCATION COUNTER TO AN ODD COUNT.
64 *
65 * ODD
66 .ODD LDA .USE.
67 STA .BFS.
68 *
69 * THE EIGHT PSEUDO OP FORCES THE LOCATION COUNTER TO AN EVEN MULTIPLE OF EIGHT.
70 * NOTICE BITS 15 THRU 17, 6TH OCTAL CHAR, OF THE LOCATION COUNTER IS ALL ZEROS
71 *
72 * EIGHT
73 .EIGHT LDA .BSS.
74 ADA .E.TC.
75 STA .EIGHT
76 *
77 * I HOPE YOU NOTICED THE NOP THAT THE ASSEMBLER INSERTED IN THE
78 * INSTRUCTION SEQUENCE WHEN ENCOUNTERING THOSE LAST THREE PSEUDO OPs.
79 *
80 *
81 * WHILE WE ARE AT IT, DID YOU NOTICE THAT THESE REMARKS IN NO WAY EFFECTED
82 * THE LOCATION COUNTER, OR THE INSTRUCTIONS.
83 *
84 *
85 *
86 * THE OCT PSEUDO OP USED HERE WILL CREATE 3 OCTAL WORDS STARTING AT LOC. .OCT.
87 *
88 * OCT. OCT 300,77777525252,-4
89 *
90 * THIS DEC PSEUDO OP WILL CREATE THREE WORDS STARTING AT LOCATION .DEC. AFTER
91 *
92 * FIRST CONVERTING FROM DECIMAL TO OCTAL.
93 *
94 * DEC. DEC -1,1.2E1032,6.
95 *
EXAMPLES OF PSUEDO OPERATIONS**

...ALSO LITERALS AND MACROS.

94 * THE BCI PSEUDO OP WILL STORE A FOUR WORD PHRASE STARTING AT LOC. .BCI.
95 *
96 .BCI. BCI 4,"I LIKE THIS PSEUDO OP.

97 * THE VFD PSEUDO OP WILL CREATE A WORD THAT IS BIT ORIENTED IN ITS EXPRESSION.
98 *
99 .VFD. VFD 18/.BCI,.06/77,.H6/-,.3/2,3/5
100 *
101 * HERE IS A REAL BEAUT. A VFD USING ALGEBRAIC AND BOOLEAN OPERATORS.
102 *
103 .ETC. VFD 4/16/2, 4 BITS OF 16 DIVIDED BY 2
104 ETC 3/2*3, 3 BITS OF 2 TIMES 3
105 ETC 5/15+3, 5 BITS OF 15 PLUS 3
106 ETC 6/17-5, 6 BITS OF 17 MINUS 5
107 ETC 06/16, BOOLEAN 6 BITS OF 16 NOT
108 ETC 03/2*3, BOOLEAN 3 BITS OF 2 AND 3
109 ETC 04/15*3, BOOLEAN 4 BITS OF 15 ORED TO 3
110 ETC 05/17-5, BOOLEAN 5 BITS OF 17 EX ORED TO 5

111 *
112 * BY THE WAY, THAT ALSO SHOWED THE USE OF THE ETC PSEUDO OP.
113 *
114 ******************************************************************************
115 *
116 * THE FOLLOWING SHOWS A USE OF THE SET AND DUP PSEUDO OPS. IT WILL PROVIDE.....
117 * CODING TO MOVE 5 WORDS FROM X TO Y.
118 *
119 .SET. SET 0
120 DUP 3,5
121 LDA X++,SET.
122 STA Y++,SET.
123 .SET. SET .SET +1
124 LDA X++,SET.
125 STA Y++,SET.
126 .SET. SET .SET +1
127 LDA X++,SET.
128 STA Y++,SET.
129 .SET. SET .SET +1
130 LDA X++,SET.
131 STA Y++,SET.
132 .SET. SET .SET +1
133 LDA X++,SET.
134 STA Y++,SET.

125 * DID YOU NOTICE THAT THE SET PSEUDO OP ALLOWED THE SAME SYMBOL IN THE LOCATION
126 * FIELD TO BE RE DEFINED WITHOUT ERROR.
EXAMPLES OF PSUEDO OPERATIONS**

...ALSO LITERALS AND MACROS.

127  EJECT
128  * THE BSS PSUEDO OP WILL RESERVE 5 WORDS STARTING AT LOCATION .BSS.
129  *
130  .BSS. BSS 5
131  *
132  * THIS PSUEDO OP WILL RESERVE 5 WORDS PRECEEDING LOCATION .BSF.
133  *
134  .BSF. BSF 5
135  *
136  * THAT LAST PSUEDO OP IS NICE IF YOU WISH TO USE REVERSE SCANNING ADDRESS
137  * MODIFIERS SUCH AS DII TALLY OR DII TALLY.
138  *
139  * THIS PSUEDO OP WILL CREATE AN INSTRUCTION WORD WITH
140  * NO OP CODE. IT IS HANDY FOR CREATING ADDRESSES FOR IR OR RI MODIFICATION.
141  *
142  .ARG. NULL
143  ARG .NULL.,7
144  ARG .DLIT.,1
145  *
146  * OR WE COULD CREATE TWO 10 BIT ADDRESSES IN ONE WORD BY USING THE
147  * FOLLOWING PSUEDO OP.
148  *
149  .ZERO. ZERO .OCT.,.DEC.
150  *
151  * BUT ON THE OTHER HAND IF YOU WOULD RATHER USE ABSOLUTE ADDRESSING RATHER
152  * THAN SYMBOLIC ADDRESSING, YOU CAN CODE IT THIS WAY.
153  *
154  ZERO 1024,512
155  *
156  * OR IF YOU CANT MAKE UP YOUR MIND, YOU CAN USE BOTH SYMBOLIC AND ABSOLUTE.
157  *
158  ZERO 1024,.ORG. MA, MA. THEY'RE BOTH THE SAME ADDRESS.
159  ************************************************************************************
160  *
161  * HERE IS AN INSTRUCTIONAL LITERAL
162  *
163  .MLIT. LDA =MLD1 5000
164  ******************************************************
165  *
166  * HERE ARE A COUPLE EXAMPLES OF VARIABLE FIELD LITERALS.
167  .VLIT. LDA =V18/START,09/253,3/0,06/17
168  LDA =V10/512,02/2,H18/82X,6/0
169  ******************************************************
170  *
171  .REF. NULL
172  REF LNRSM LIST NONREFERENCED SYMBS (EGI .REF.)
173  NOTICE: THIS WILL CAUSE A COMPLETE REFERENCE LISTING OF ALL THE MME'S.
EXAMPLES OF PSEUDO OPERATIONS**

...ALSO LITERALS AND MACROS.

174 * HERE ARE SOME EXAMPLES OF DECIMAL LITERALS.
175 *

176 .LIT.        LDA    =10 ALL
177 .LDA        LDA    =-15 NEGATIVE DECIMAL INTEGER
178 .LDA        LDA    =6.0 SINGLE PRECISION FLOATING POINT
179 .LDA        LDA    =1.55E1 SINGLE PRECISION FLOATING POINT
180 .LDA        LDA    =5.1202 DOUBLE PRECISION FLOATING POINT
181 .LDA        LDA    =5017 SINGLE PRECISION FIXED POINT
182 .LDA        LDA    =22.585 SINGLE PRECISION FIXED POINT
183 .LDA        LDA    =1.2E1832 SINGLE PRECISION FIXED POINT
184 .LDA        LDA    =1.9501837 DOUBLE PRECISION FIXED POINT

185**********************************************************************

002104 .LIT.        NULL  000 000
002105 .LIT.        LIT    000 000 CAUTION...NOTICE THE LOCATION COUNTER.

188**********************************************************************

189 *
190 * IF I USE THE SYMREF PSEUDO OP, I CAN REFERENCE A SYMBOL WHICH DOES NOT......
191 * APPEAR IN THIS ASSEMBLY, AND WILL NOT CAUSE AN ERROR.
192 *

193 SYMREF .SYMREF
194 TRA .SYMREF

195 *
196 * OF COURSE IT IS EXPECTED THAT THIS SAME SYMBOL WILL HAVE BEEN DEFINED.......
197 * BY A SYMDEF IN ANOTHER ASSEMBLY TO BE EXECUTED WITH THIS ONE.
198 **********************************************************************

199 *
200 * IF I NEED TO TURN BIT 28 ON, I WOULD USE THE FOLLOWING PSEUDO OP.
201 *
202 INHIB ON

203 INHIB LDA START

204 STA .INHIB

205 *
206* TO TURN IT OFF, I MERELY DO THE FOLLOWING.
207 *
208 INHIB OFF

209 LDA START

210 STA .INHIB

211 *
EXAMPLES OF PSEUDE OPERATIONS**

...ALSO LITERALS AND MACROS.

212 * THE FOLLOWING IS AN EXAMPLE OF THE USE OF THE CALL, SAVE, AND RETURN PSEUDOS.
213 * NOTE: THE SAVE AND RETURN PSEUDO OPS MUST APPEAR IN THE ASSEMBLY BEFORE....
214 * THE CALL PSEUDO OP.
215 *

002130
02130 002132710000 010
02131 00220463000 010
02132 002204756000 010
02133 002204741000 010
02134 002204 2350 00 010
02135 002137 7550 00 010
02136 002131710000 010

216 .SAVE SAVE THE SYMBOL .SAVE WILL APPEAR AS A SYMDEF.

217 LDA .USE. BODY OF THE SUBROUTINE
218 STA .CALL DITTO
219 RET. RETURN .SAVE

220 *
221 * NOTE ** THE CALL PSEUDO OP MUST FOLLOW THE SAVE AND RETURN PSEUDO OPS........
222 * IF USED IN THE SAME ASSEMBLY.
223 *
224 *

002137
02137 002130710000 010
02140 002132710000 010
02141 002204600300 010
02142 002104 6350 00 010

225 CALL CALL .SAVE THE SYMBOL .SAVE WILL APPEAR AS A SYMREF.

226 EAA .LIT. RETURN POINT INTO PROGRAM.

227 *
228 ....................................................................................
229 *
230 * HERE ARE TWO EXAMPLES OF FLOATABLE CODE..........
231 *

002143
02143 777637 2350 04 000
02144 777636 2350 04 0002 2002

232 .FLOAT LDA START=.IC AS YOU CAN SEE, FLOATABLE CODE....
233 LDA START=. MAY BE CODED EITHER WAY.
234 *
235 * SUPPOSE YOU WOULD PREFER TO USE THE MNEMONICS LDAR INSTEAD OF LDA.....
236 * FOR THE LOAD THE A REGISTER INSTRUCTION. THEN YOU WOULD USE THE FOLLOWING.
237 *
238 LDAR OPSYN LDA
239 *
240 * NOW LETS SEE WHAT HAPPENS WHEN WE USE IT.
241 *

002145
02145 002070 2350 00 010
02146 002071 2350 00 010
02147 002202 7010 00 010

242 .OPSYN LDAR .MLIT.
243 LDA .VLIT.
244 TSX .USE.
245 *
246 * WATCH ME CHANGE THE LOCATION COUNTER BY USING THE USE PSEUDO OP.
247 *

002202
02202 002143 2350 00 010
002203
02203 002136 7550 00 010

248 USE NEXT NEXT IS THE NAME OF MY NEW LOCATION CTR.
249 .USE LDA .FLOAT
250 STA .RET.
251 *
252 * NOW WATCH ME PICK UP WHERE I LEFT OFF WITH THE ORIGINAL LOCATION COUNTER.
253 *
254 *
255 USE
EXAMPLES OF PSUEDO OPERATIONS

...ALSO LITERALS AND MACROS.

002150 000000 2350 00 020
002151 010000 0750 00 030
002152 000000 7550 00 020

256 * SUPPOSE YOU WANT TO SHOW BLANK OR LABELLED COMMON AREAS, BELOW THE CODING.
257 LDA .BLOCK
258 ADA WW
259 STA X
260 *
261 * THE FOLLOWING BSS PSUEDO OPS WILL RESERVE AREAS IN BLANK COMMON.
262 * NOTE*** INSTRUCTIONS OR DATA MAY NOT BE PLACED IN BLANK COMMON.
263 *
264 BLOCK THIS IS BLANK COMMON OF COURSE.
265 .BLOCK BSS 5
266 X BSS 5
267 Y BSS 5
268 Z BSS 5
269 *
270 *
271 BLOCK LABCOM AND THIS IS LABELED COMMON.
272 WW BSS 10
273 XX DEC 99
274 YY OCT 77
275 ZZ BCI 4, THIS IS LABELED COMMON.
276 *
277 * NOTICE THAT LABELED COMMON PERMITTED THE USE OF CONSTANTS.....
278 * WITHIN THE LABELED COMMON AREA.
279 *
280 USE AGAIN I RETURN TO MY ORIGINAL COUNTER
281 *
282 * HERE ARE EXAMPLES OF THE FOUR TALLY PSUEDO OPS.
283 *
284 *
002153 020000 2350 52 030
002154 020000 2360 52 030
002155 020000 2350 57 030
002156 020000 2360 53 030

285 LDA .TALY,,SC
286 LDO .TALD,,SC
287 STA .TALC,,IDC
288 STQ .TALD,,AD
289 *
290 *
291 BLOCK TALLY'S LABELED COMMON AGAIN.
292 .TALY, TALLY WW,10,2
293 .TALD, TALLYB WW,10,1
294 .TALC, TALYIC YY,10,7
295 .TALD, TALLY ZZ,10,10
296 *
297 USE BACK TO THE ORIGINAL LOCATION CTR AGAIN.
298
EXAMPLES OF PSEUDO OPERATIONS

...ALSO LITERALS AND MACROS.

299         EJECT
300 *        REFMA ON
301          CREATES A SEPERATE MACRO SYMBOL TABLE.
302 **********************************************************
303 *        THIS NEXT SEQUENCE OF INSTRUCTIONS WILL SERVE TO EXPLAIN THE USE OF MACROS.
304 *        IT WILL ALSO SHOW THE USE OF THE CONDITIONAL PSEUDO OPS, CREATED SYMBOLS,
305 *        IDRP AND THE PRINT MACRO EXPANSION ON/OFF PSEUDO OP.
306 *        PMC ON
307 *        PMC ON
308 *        TURN PMC ON TO SEE EXPANDED CODE
309 *        ...
310 *        ...
311 *        ...
312 **********************************************************
313 *
314 HOLY MACRO
315 LDX7 #1,DU
316 LDX6 0,DU
317 &2 LEA #3,6
318 STA #4,6
319 IFG #1,1,3
320 *
321 ADX6 1,DU
322 SBX7 1,DU
323 TNZ #2
324 ENDM HOLY
325 *
326 * ARGUMENT NUMBER 2 IS IMPLICITLY NULLED TO FORCE CREATED SYMBOLS.
327 HOLY 25,#x,y
328 LDX7 25,DU
329 LDX6 0,DU
330 .001.
331 LDA X,6
332 STA Y,6
333 IFG 25,1,3
334 ADX6 1,DU
335 SBX7 1,DU
336 TNZ .001.
337 *
338 * HERE WE USED THE SYMBOLIC LOCATION BIG RATHER THAN CREATED SYMBOLS.
339 HOLY 1,BIG,XX,YY
340 LDX7 1,DU
341 LDX6 0,DU
342 BIG LDA XX,6
343 STA YY,6
344 IFG 1,1,3
345 *
346 * LETS SEE THAT LAST ONE WITH PMC OFF THIS TIME.
347 PMC OFF
348 *
349 HOLY 1,BIGGER,XX,YY
EXAMPLES OF PSUEDO OPERATIONS

...ALSO LITERALS AND MACROS.

335   EJECT
336 *
337 * WATCH CAREFULLY NOW. HERE IS THE IDRP PSUEDO OP AT WORK. WITH THIS
338 * WE SHALL SEE THE ARG PSUEDO OP AGAIN.
339 DOIT MACRO
340   TSXI #1
341   IDRP #2 REPEAT NEXT INST BY NUMBER OF SUB ARG#S IN ARG#2.
342   ARG #2   THIS IS THE ARG PSUEDO OP.
343   IDRP   THIS IDRP IS USED AS A DELIMITER.
344   ENDM DOIT
345 *
346 * LET'S TURN PMC ON AND SEE THIS MACRO EXPANDED.
347 *
348   PMC ON
349 *
350 DOIT START,(XX,YY,ZZ)
351   TSXI START
352   IDRP XX,YY,ZZ
353   ARG XX
354   ARG YY
355   ARG ZZ

356 ************************************************************
357 *
358 * TH-TH-THATS ALL FOLKS, BUT LET US NOT FORGET THE END CARD.
359 *

ERROR LINKAGE

350 DOIT END
002204 0000000000000 000
002205 33624242526 000

2206 IS THE NEXT AVAILABLE LOCATION. GMAP VERSION JMPA/06970 JMPB/06970 JMPC/06970
THERE WERE NO WARNING FLAGS IN THE ABOVE ASSEMBLY
<table>
<thead>
<tr>
<th>DECAL</th>
<th>SYMBOL</th>
<th>REFERENCES BY ALTER NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2170</td>
<td>BIG</td>
<td>330 330</td>
</tr>
<tr>
<td>2174</td>
<td>BIGGER</td>
<td>334 334</td>
</tr>
<tr>
<td>10</td>
<td>GEBO</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>GECA</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>GECH</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>GEED</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>GEFA</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>GEFO</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>GEOF</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>GEFI</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>GEFRC</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>GEFSE</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>GEGIN</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>GELAPS</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>GELBAR</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>GELOC</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>GEMOR</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>GEMREL</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>GENME</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>GEPRIO</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>GERE</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>GERELS</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>GERETS</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>GERAD</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>GEROLL</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>GEROUT</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>GERSTR</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>GESA/E</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>GESET</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>GESNA</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>GESNUM</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>GESPEC</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>GESYOT</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>GETIME</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>GEUSER</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>GEMAKE</td>
<td></td>
</tr>
<tr>
<td>2161</td>
<td>.001</td>
<td>327 327</td>
</tr>
<tr>
<td>2063</td>
<td>.ARG</td>
<td>142 54 142</td>
</tr>
<tr>
<td>2031</td>
<td>.BBI</td>
<td>96 96 96</td>
</tr>
<tr>
<td>2063</td>
<td>.BFS</td>
<td>134 67 134</td>
</tr>
<tr>
<td>0</td>
<td>.BLCK</td>
<td>265 257 265</td>
</tr>
<tr>
<td>1200</td>
<td>.BOOL</td>
<td>49 49 50</td>
</tr>
<tr>
<td>2091</td>
<td>.BS</td>
<td>130 73 130</td>
</tr>
<tr>
<td>2137</td>
<td>.CALL</td>
<td>225 218 225</td>
</tr>
<tr>
<td>2026</td>
<td>.DEC</td>
<td>92 92 149</td>
</tr>
<tr>
<td>2038</td>
<td>.BLIT</td>
<td>176 144 176</td>
</tr>
<tr>
<td>2020</td>
<td>.EIGHT</td>
<td>73 73 75</td>
</tr>
<tr>
<td>2204</td>
<td>.E.L.</td>
<td>55 216 225</td>
</tr>
</tbody>
</table>
### OCTAL SYMBOL REFERENCES BY ALTER NO.

<table>
<thead>
<tr>
<th>OCTAL</th>
<th>SYMBOL</th>
<th>REFERENCES</th>
<th>ALTER NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>.EMM</td>
<td>26 36 37</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>.EOU</td>
<td>103 74 103</td>
<td></td>
</tr>
<tr>
<td>2036</td>
<td>.ETC</td>
<td>60 22 60</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>.EVM</td>
<td>232 232 249</td>
<td></td>
</tr>
<tr>
<td>2143</td>
<td>.FLOAT</td>
<td>203 203 204</td>
<td>210</td>
</tr>
<tr>
<td>2124</td>
<td>.INHIA</td>
<td>186 186 226</td>
<td></td>
</tr>
<tr>
<td>2104</td>
<td>.LIT</td>
<td>163 163 242</td>
<td></td>
</tr>
<tr>
<td>2070</td>
<td>.MLIT</td>
<td>53 53 143</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>.NULL</td>
<td>88 88 149</td>
<td></td>
</tr>
<tr>
<td>2023</td>
<td>.OCT</td>
<td>66 61 66</td>
<td></td>
</tr>
<tr>
<td>2015</td>
<td>.OOO</td>
<td>242 32 242</td>
<td></td>
</tr>
<tr>
<td>2145</td>
<td>.PSYN</td>
<td>31 31 158</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>.ORG</td>
<td>171 171</td>
<td></td>
</tr>
<tr>
<td>2073</td>
<td>.REF</td>
<td>219 219 250</td>
<td></td>
</tr>
<tr>
<td>2136</td>
<td>.RET</td>
<td>216 216 219</td>
<td>225</td>
</tr>
<tr>
<td>2130</td>
<td>.SAVE</td>
<td>123 119 121</td>
<td>122 123</td>
</tr>
<tr>
<td>5</td>
<td>.SET</td>
<td>32 21 32</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>.SMDEF</td>
<td>193 194</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>.SMREF</td>
<td>293 296 293</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>.TALB</td>
<td>294 287 294</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>.TALC</td>
<td>295 288 295</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>.TALD</td>
<td>292 285 292</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>.TALY</td>
<td>249 66 217</td>
<td>244 249</td>
</tr>
<tr>
<td>2207</td>
<td>.USE</td>
<td>99 68 99</td>
<td></td>
</tr>
<tr>
<td>2635</td>
<td>.VFD</td>
<td>167 167 243</td>
<td></td>
</tr>
<tr>
<td>2071</td>
<td>.VLIT</td>
<td>149 33 149</td>
<td></td>
</tr>
<tr>
<td>2059</td>
<td>.ZERO</td>
<td>33 33 43</td>
<td>44 167 203 209 232 233 350</td>
</tr>
<tr>
<td>2002</td>
<td>.START</td>
<td>43 43 45</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>WW</td>
<td>272 256 272</td>
<td>292</td>
</tr>
<tr>
<td>12</td>
<td>X</td>
<td>266 121 123</td>
<td>259 266 327</td>
</tr>
<tr>
<td>12</td>
<td>XX</td>
<td>273 273 293</td>
<td>330 334 350</td>
</tr>
<tr>
<td>12</td>
<td>Y</td>
<td>267 122 123</td>
<td>267 327</td>
</tr>
<tr>
<td>13</td>
<td>YY</td>
<td>274 274 294</td>
<td>330 334 350</td>
</tr>
<tr>
<td>17</td>
<td>Z</td>
<td>268 268 268</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>ZZ</td>
<td>275 275 295</td>
<td>350</td>
</tr>
<tr>
<td>OCTAL</td>
<td>SYMBOL</td>
<td>REFERENCES BY ALTER NO.</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>------------------------</td>
<td></td>
</tr>
<tr>
<td>2176</td>
<td>DOIT</td>
<td>339 350</td>
<td></td>
</tr>
<tr>
<td>2157</td>
<td>HOMY</td>
<td>314 327 330 334</td>
<td></td>
</tr>
</tbody>
</table>

**17845 WORDS OF MEMORY WERE USED BY GMAP FOR THIS ASSEMBLY.**
MACROS

MACRO PROTOTYPE:

1     8
DOIT  MACRO               (HEADING)

LDA   = 5
ADA   COUNT
STA   TOTAL

)}

ENDM   DOIT               (TERMINATOR)

MACRO EXPANSION CALL:

DOIT
MACRO

WITH

SUBSTITUTABLE ARGUMENTS

MACRO PROTOTYPE:

DOIT . MACRO
LDA #1
ADA #2
STA #3
ENDM DOIT

MACRO CALL:

DOIT COUNT,=5,TOTAL
REFRESHING TALLY WORDS

WITH MACRO CALL

<table>
<thead>
<tr>
<th>RESTOR</th>
<th>MACRO</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAX6</td>
<td>#2</td>
</tr>
<tr>
<td>EAX7</td>
<td>#3</td>
</tr>
<tr>
<td>RPD</td>
<td>#1.1</td>
</tr>
<tr>
<td>LDA</td>
<td>0.6</td>
</tr>
<tr>
<td>STA</td>
<td>0.7</td>
</tr>
<tr>
<td>ENDM</td>
<td>RESTOR</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>RESTOR</td>
<td>150, TALSAV, TALWRK</td>
</tr>
</tbody>
</table>

THIS EXAMPLE WILL RESTORE UP TO
256 TALLY WORDS WITH ONE CALL
### LOCATION FIELD SYMBOLS

<table>
<thead>
<tr>
<th>1</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOLY</td>
<td>MACRO</td>
<td></td>
</tr>
<tr>
<td>GO</td>
<td>EAX7</td>
<td>0</td>
</tr>
<tr>
<td>LDA</td>
<td>STA</td>
<td>#1,7</td>
</tr>
<tr>
<td>ADX7</td>
<td>#2,7</td>
<td></td>
</tr>
<tr>
<td>CMPX7</td>
<td>1,DU</td>
<td></td>
</tr>
<tr>
<td>TNZ</td>
<td>#3,DU</td>
<td></td>
</tr>
<tr>
<td>ENDM</td>
<td>GO</td>
<td>HOLY</td>
</tr>
</tbody>
</table>
NESTING OF MACROS

PLEASE MACRO

TZE #1
TRC #2
ENDM PLEASE

AGAIN MACRO

CMPA #1
PLEASE #2
ENDM AGAIN

DOIT MACRO

LDA #1
AGAIN #2
STA #3
ENDM DOIT
MACRO WITH CONDITIONAL PSEUDO OP

MACRO Prototype:

<table>
<thead>
<tr>
<th>BINBCD</th>
<th>MACRO</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAX7</td>
<td>0</td>
</tr>
<tr>
<td>LDA</td>
<td>#1</td>
</tr>
<tr>
<td>EAQ</td>
<td>0</td>
</tr>
<tr>
<td>IFE</td>
<td>#1,SEQBIN,3</td>
</tr>
<tr>
<td>RPT</td>
<td>3.1</td>
</tr>
<tr>
<td>BCD</td>
<td>TAF1.7</td>
</tr>
<tr>
<td>STQ</td>
<td>SEQBCD</td>
</tr>
<tr>
<td>INE</td>
<td>#1,SEQBIN,6</td>
</tr>
<tr>
<td>RPT</td>
<td>4.1</td>
</tr>
<tr>
<td>BCD</td>
<td>TAB.7</td>
</tr>
<tr>
<td>STQ</td>
<td>#2</td>
</tr>
<tr>
<td>RPT</td>
<td>6.1</td>
</tr>
<tr>
<td>BCD</td>
<td>0.7</td>
</tr>
<tr>
<td>STQ</td>
<td>#2+1</td>
</tr>
<tr>
<td>ENDM</td>
<td>BINBCD</td>
</tr>
</tbody>
</table>

MACRO CALL:

- BINBCD
- EBIN, EBCD
- BINBCD
- FBIN, FBCD
- BINBCD
- SEQBIN
MACRO WITH
INDEFINITE REPEAT

1

DOIT

8

MACRO

LDA #1

IDRP #2

ADA #2

IDRP

STA #3

ENDM DOIT
MACRO WITH SET
AND INDEFINITE REPEAT

1

XFER
K

8

MACRO
SET
IDRP

16

0
#2
K+1

XFER
SET
IDRP
TSX1
TRA
IDRP
ARG
IDRP
ENDM

XFER
#1
*+1+K
#2
#2
XFER
EXTERNAL INTERRUPTS

PROCESSOR

SYSTEM CONTROLLER

IOM

TERMINATE

INITIATE

SPECIAL
INTERNAL INTERRUPTS
(FAULTS)

PROGRAM GENERATED

INSTRUCTION GENERATED

HARDWARE GENERATED

PROCESSOR

SYSTEM CONTROLLER

IOM
LEVELS OF I/O

WRITE

CALL PUT

MME GEINOS

CONNECT

(E.G., COBOL)

FILE AND RECORD CONTROL
(GEFC) SUBROUTINE

(REQUEST FOR I/O)

(IOS)
Pages 26-171 through 26-182 have been deleted from this handbook.
TAPE FILE CARD

ASSIGN A TAPE UNIT TO THE PROGRAM.

TAPE AB, AIS, 24,0001, BIGFILE

FILE NAME

REEL SEQUENCE NUMBER

FILE SERIAL NUMBER

SECONDARY LOGICAL UNIT DESIGNATOR

LOGICAL UNIT DESIGNATOR

FILE CODE
MASS STORAGE FILE CARD

ALLOCATE A TEMPORARY FILE ON SYSTEM MASS STORAGE.

$ FILE AC,X11S,5L,DS180

DEVICE TYPE DESIRED

ACCESS

LOGICAL UNIT DESIGNATOR

FILE CODE
SYSOUT & DATA FILE CARDS

ASSIGN OUTPUT FILES TO SYSTEM OUTPUT MEDIA CONVERSION.

$  SYSOUT  AD
    FILE CODE

WRITE FILES onto TEMPORARY LINKED SYSTEM STORAGE FOR INPUT INTO USER PROGRAM.

$  DATA  AE
    FILE CODE
MEDIA CONVERSION

$ DATA IN

$ SNUMB
$ IDENT

$ GMAP

$ END

$ EXECUTE

$ DATA IN

$ SYSOUT PN

$ SYSOUT PN
$ SYSOUT PR

$ SYSOUT PR

$ ENDJOB
LOGICAL AND PHYSICAL RECORDS

LOGICAL RECORD

IRG

BLOCK SERIAL NUMBER

RECORD SIZE

MEDIA CODE

REPORT CODE

DATA

PHYSICAL RECORD

IRG

DATA

DATA

DATA

DATA

E O F

E O F
VARIABLE LENGTH RECORDS

<table>
<thead>
<tr>
<th>0</th>
<th>17</th>
<th>28</th>
<th>29</th>
<th>30</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECORD SIZE</td>
<td>MEDIA CODE</td>
<td>REPORT CODE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIXED LENGTH RECORDS

| DATA |
PARTITIONED RECORDS

0  17  24  25  26  29  30  35
RECORD SIZE  SEG. CODE  MEDIA REPORT CODE
DATA
FIRST SEGMENT 01

0  17  24  25  26  35
SEGMENT SIZE  SEG. CODE  SEGMENT NUMBER
DATA
INTERMEDIATE SEGMENTS 10

0  17  24  25  26  35
SEGMENT SIZE  SEG. CODE  SEGMENT NUMBER
DATA
LAST SEGMENT 11
$ GMAP

FILWD VFD 18/ MYFILE, 18/0

FILCB MYFILE AA, BUFFT

BUFFT BSS 321

$ EXECUTE

$ DATA AA

$ ENDJOB
FILE DESIGNATOR WORD

VFD

18/MYFILE,1/1,17/0

0

MYFILE

1 0 0 0 0 0 0 0 0

0

SYMBOLIC NAME OF
FILE CONTROL BLOCK

FILE NUMBER

REQUIRED FILE
OPTION

PUT SIZE ON OUTPUT
OPTION

PRIME INPUT BUFFER
OPTION

REWIND ON CLOSE

REWIND ON OPEN

INPUT/OUTPUT INDICATOR
OPEN FILES

CALL OPEN(AWD, 2)

VFD 18/FILEAA, 18/0
VFD 18/FILEBB, 1/1, 17/0

FILCB FILEAA, AA, BUFFA1, BUFFA2

FILCB FILEBB, BB, BUFFB1

BUFFA1 BSS 321

$ DATA AA

$ TAPE BB
CLOSE FILES

CALL CLOSE(AWD, 2)

AWD VFD 18/FILEAA, 18/0
VFD 18/FILEBB, 1/1, 17/0

FILCB FILEAA, AA, BUFFA1, BUFFA2

FILCB FILEBB, BB, BUFFB1

BUFFB1 BSS 321

$ DATA AA

$ TAPE BB
GET LOGICAL RECORD

CALL GET (FILEAA, ENDAA, WSA)

FILCB FILEAA, AA, BUFFA1, BUFFA2

BUFFA1 BSS 321

WSA BSS 14

⇒
WRITE LOGICAL RECORD

CALL PUT (FILEBB, WSA)
FILCB FILEBB, BB, BUFFB1

LOC-SYM +1 Record Size

BUFFB1 BSS 321

WSA BSS 14
INITIALIZE EDITING PARAMETERS

CALL IOEDIT(LLIST, 5)

LLIST ZERO HEAD1, 1
ZERO HEAD2, 2
ZERO PCHCD, 3
ZERO RPCD, 7
ZERO PGNO, 5

HEAD1 BCI 10 WORDS

39686 01 03-16-70 19.268
31 \( \downarrow \) \( \downarrow \) 90

10 WORDS

HEAD2 BCI

PCHCD BCI 2, GAR00100

GAR00102
GAR00101
GAR00100

73 - 80

20 WORDS
WRITE LOGICAL RECORD

FILCB OUTFL EE, OTBUFF

LOC-SYM
+1 Record Size

CALL WTREC (OUTFL, WSEET, <=3, <=20)

OTBUFF BSS 321

WSEET BSS 30
WRITE LOGICAL RECORD

FILCB (FLFF, FF, BUFFF)

LOC-SYM +1 Record Size

CALL PRINT (FLFF, FFWS = 2)

BUFFF BSS 321

FFWS BCI 22
WRITE LOGICAL RECORD

CALL PUNCH(FLFF,BINWS,=0)

FILCB FLFF,FF,BUFFF

BUFFF BSS 321

BINWS BCI 24
PHYSICAL RECORD PROCESSING
SEQUENTIAL FILES

FILCB PHINS SI
CALL READ (PHINS, CNTA, INSCC)

CNTA
IOTP AREA3,25
IOTP AREA1,50
IOTD AREA2,50

AREA1 BSS 100

AREA2 BSS 50

AREA3 BSS 25
PHYSICAL RECORD PROCESSING
RANDOM FILES

CALL READ(PHINR,CNTC)

DEC 35
IOTP AREA8,40
IOTP AREA7,80
IOTD AREA9,40

FILCB PHINR,RI

CNTC

$ DISC RI,X1S,1R

AREA7 BSS 80

AREA8 BSS 40

AREA9 BSS 40
## FILE CONTROL BLOCK

**FILCB MYFILE,AA,BUFF1**

<table>
<thead>
<tr>
<th>LOC-SYM</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>-14</td>
<td>PREHEADER LABEL EXIT</td>
</tr>
<tr>
<td>-13</td>
<td>POSTHEADER LABEL EXIT</td>
</tr>
<tr>
<td>-12</td>
<td>PRETRAILER LABEL EXIT</td>
</tr>
<tr>
<td>-11</td>
<td>POSTTRAILER LABEL EXIT</td>
</tr>
<tr>
<td>-10</td>
<td>FIRST SIX CHARACTERS OF FILE NAME</td>
</tr>
<tr>
<td>-9</td>
<td>LAST SIX CHARACTERS OF FILE NAME</td>
</tr>
<tr>
<td>-8</td>
<td>RETENTION PERIOD</td>
</tr>
<tr>
<td>-7</td>
<td>FILE SERIAL NUMBER</td>
</tr>
<tr>
<td>-6</td>
<td>BLOCK COUNT</td>
</tr>
<tr>
<td>-5</td>
<td>ERROR ROUTINE EXIT</td>
</tr>
<tr>
<td>-4</td>
<td>PAT POINTER</td>
</tr>
<tr>
<td>-3</td>
<td>IOS STATUS RETURN WORD</td>
</tr>
<tr>
<td>-2</td>
<td>IOS STATUS RETURN WORD</td>
</tr>
<tr>
<td>-1</td>
<td>FCB POINTER</td>
</tr>
<tr>
<td></td>
<td>PHYSICAL DEVICE ADDRESS</td>
</tr>
<tr>
<td>+1</td>
<td>RECORD SIZE</td>
</tr>
<tr>
<td>+2</td>
<td>LOCATION OF FIRST BUFFER</td>
</tr>
<tr>
<td>+3</td>
<td>GFRC WORKING STORAGE WORD</td>
</tr>
<tr>
<td>+4</td>
<td>BLOCK SIZE</td>
</tr>
<tr>
<td>+5</td>
<td>RECORD SIZE ROUTINE LOCATION</td>
</tr>
</tbody>
</table>

## FILE DESIGNATOR WORD

**FWD VFD 18/MYFILE,18/0**

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>MYFILE</td>
</tr>
</tbody>
</table>
FILE CONTROL BLOCK MACRO EXAMPLE

1  TTL  BASIC SA COURSE - GEFRC EXAMPLES
2  TIL  FILE CONTROL BLOCK MACRO EXAMPLE
3  SYMDEF  GC
4  WJ
5  *
6  *
7  *
8  Example of complete file control
9  block macro-instruction
10  *

11  JSE  COUNT
12  FILEB  MYFILE;
13  STC  22,
14  STC  BUFF1;
15  STC  320,
16  STC  0,
17  STC  0
18  STC  0
19  STC  0
20  STC  0
21  STC  0
22  STC  0
23  STC  0
24  STC  0
25  STC  0
26  STC  0
27  STC  0
28  STC  0
29  STC  0
30  STC  MYFILE,
31  STC  0,
32  STC  0

SYMBOLIC NAME OF FILE CONTROL BLOCK
FILE CODE
FIRST BUFFER
SECOND BUFFER
MAXIMUM BLOCK SIZE
RECORD FORM
RECORD SIZE
NO BLOCK SERIAL NUMBERS
ERROR ROUTINE
NO STANDARD LABELS
RECORDING MODE
RECORDING DENSITY
MULTIPLE FILE REEL
RETENTION PERIOD
PREHEADER LABEL ROUTINE
POSTHEADER LABEL ROUTINE
PRETRAILER LABEL ROUTINE
POSTTRAILER LABEL ROUTINE
FILENAME
GEPR OVERRIDE
PARTITIONED RECORD
FILE DESIGNATOR WORD AND BUFFERS

35.
36.
37. EXAMPLE OF FILE DESIGNATOR WORD
38.
39.
40 ЗЗПД VPD 18/MYPFILE, SYMBOLIC NAME OF FILE CONTROL BLOCK
41. ETC 1/1; INPUT/OUTPUT INDICATOR
42. ETC 1/0; REWIND ON OPEN
43. ETC 2/0; REWIND ON CLOSE
44. ETC 1/0; PRIME INPUT BUFFER OPTION
45. ETC 1/0; PUT SIZE ON OUTPUT OPTION
46. ETC 1/0; REQUIRED FILE OPTION
47. ETC 11/0; FILE POSITIONING VALUE
48.
49.
50. RESERVE BUFFER SPACE:
51.
52. EACH BUFFER MUST BE ONE WORD LARGER THAN
53. THE PHYSICAL RECORD FOR A BUFFER CONTROL WORD
54. TO CONTROL THE DATA FLOW TO OR FROM THE BUFFER
55.
56.
57 RUFF1 5SS 321; FIRST BUFFER
58 RUFF2 5SS 321; SECOND BUFFER
BASIC SA COURSE: * GEFR Examples:

CALL: OPEN, CLOSE, GET, PUT MACRO-INSTRUCTIONS.

61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104

THE FOLLOWING IS A DESCRIPTION OF A FILE CONTROL CARD
FOR A CARD FILE USING A $DATA FILE, CONTROL CARD.
BECAUSE OF THE USE OF THE $DATA $FILE CONTROL CARD THIS FILE WILL BE IN STANDARD SYSTEM FORMAT.

FILECB FILEAA, AA, BUFFA2, BUFFA2

FILE CONTROL BLOCK FOR OUTPUT FILE CODE BB
USING STANDARD SYSTEM FORMAT WITH ONE BUFFER
THIS IS A TAPE FILE USING A $TAPE $BB $FILE CONTROL CARD

FILECB FILEBB, BB, BUFFB1

FILE DESIGNATOR WORDS ASSOCIATED WITH FILEAA AND FILEBB

BUFFERS FOR FILEAA AND FILEBB

BUFFA1 3SS 321 FIRST BUFFER FOR FILEAA
BUFFA2 3SS 321 SECOND BUFFER FOR FILEAA
BUFFB1 3SS 321 BUFFER FOR FILEBB

WORKING STORAGE AREA INTO WHICH GEFR WILL MOVE THE LOGICAL RECORD

WSA 3SS 14
JSE
CALL, OPEN, CLOSE, GET, PUT MACRO INSTRUCTIONS

107
108
109 INITIALIZE TWO FILES BEGINNING AT A LIST OF FILE DESIGNATOR WORDS LOCATED AT AWD
111
112
113 CALL OPEN(AWD, 2)

000001 0400006719900 030
000002 0000067199900 010
000003 007210000161 010
000004 001502009900 010
000005 000002009900 000
CALL OPEN, CLOSE, GET, PUT MACRO INSTRUCTIONS

116
117
118  GET THE NEXT LOGICAL RECORD FROM THE
119  BUFFER OF FILEAA AND MOVE IT TO WORKING
120  STORAGE AREA WSA
121
122
123  CALL GET(FILEAA, ENDA, WSA)

124
125
126  GET THE FIRST LOGICAL RECORD IN THE NEXT
127  PHYSICAL RECORD FROM FILEAA AND MOVE IT TO WORKING STORAGE AREA WSA
128
129
130
131  CALL GETBK(FILEAA, ENDA, WSA)

000006 0100007010000 030
000007 000019100000 010
000010 0072100010173 010
000011 001530000000 010
000012 000052000000 010
000013 003457000000 010

000014 100000701800 030
000015 000027100000 010
000016 007210001203 010
000017 001530000000 010
000020 000052000000 010
000021 003457000000 010
CALL: OPEN, CLOSE, GET, PUT MACRO-INSTRUCTIONS

134 *
135 *
136 * TO INSERT THE NEXT LOGICAL RECORD INTO THE
137 * NEXT AVAILABLE SPACE IN THE OUTPUT BUFFER
138 *
139 * THE SIZE OF THE LOGICAL RECORD MUST
140 * BE PLACED IN WORD PLUS 1 OF THE
141 * FILE CONTROL BLOCK WHEN CALL PUT MACRO IS USED.
142 *
143 * SEVERAL METHODS CAN BE USED
144 * TO PLACE A BINARY VALUE IN THE
145 * RECORD SIZE OF THE FILE CONTROL BLOCK
146 *
147 * (1.) USING A LITERAL
148 *
149 *
150 LDX6 #14,DU
151 STX6 FILEBB+1
152 *
153 *
154 (2.) OR USING THE CONTENTS OF ANOTHER FIELD
155 *
156 *
157 LDX6 FOR
158 STX6 FILEBB+1
159 *
160 *
161 JSE COUNT
162 FOR VFD 18/14,18/0
163 *
164 *
165 *
166 INSERT THE NEXT LOGICAL RECORD FROM AREA WSA INTO
167 THE NEXT AVAILABLE SPACE OF THE OUTPUT BUFFER
168 *
169 *
170 CALL PUT(FILEBB,WSA)
171 *
172 *
173 TO INSERT THE NEXT LOGICAL RECORD AS THE FIRST LOGICAL
174 * RECORD OF THE NEXT PHYSICAL RECORD FROM AREA WSA
175 *
176 *
177 CALL PJTOK(FILEBB,WSA)
CALL OPEN, CLOSE, GET, PUT MACRO-INSTRUCTIONS
CALL OPEN, CLOSE, GET, PUT MACRO-INSTRUCTIONS

180
181
182
183
184
185
186
187

THIS SAME OPERATION COULD HAVE BEEN
ACCOMPLISHED EASIER USING THE CALL COPY MACRO

CALL GET(FILEAA, EVDAA)

CALL COPY(FILEBB, FILEAA)
CALL OPEN, CLOSE, GET, PUT MACRO=INSTRUCTIONS

190
191
192 THIS IS THE END-OF-FILE ROUTINE WHERE
193 CONTROL WILL BE TRANSFERRED WHEN ALL
194 OF THE INPUT RECORDS HAVE BEEN PROCESSED
195 AND A 17 OCTAL END-OF-FILE MARK IS ENCOUNTERED
196
197 THE CALL CLOSE MACRO DISCONNECTS THE FILES
198 SPECIFIED BY THE FILE DESIGNATOR WORDS
199
200
201 ENDA CALLCLOSE (AMD, 2)

202 **ME GEFINI
PROCESSING FIXED LENGTH RECORDS

205 *
206 *
207 * THIS EXAMPLE SHOWS HOW TO PROCESS FIXED
208 * LENGTH RECORDS AND MOVE LOGICAL RECORDS
209 * FROM FILECC TO FILECD
210 *
211 * FILE CONTROL BLOCK, FILE DESIGNATOR WORD,
212 * BUFFER, AND WORKING STORAGE AREA FOR
213 * INPUT FILECC USING A $ TAPE CC
214 *
215 * FILE CONTROL CARD
216 *
217 003476 JSE COUNT
218 003476 FILEB FILECC,CC,BUFFC1,,101,1;20
219 003513 003506000000 010
220 003514 FILED 19/FILCC,19/0
221 003662 BURF 35S 102 BUFFER FOR FILECC
222 *
223 *
224 * FILE CONTROL BLOCK, FILE DESIGNATOR WORD,
225 * AND BUFFER FOR OUTJPY FILE DD USING
226 * A $ TAPE DD CONTROL CARD
227 *
228 *
229 003706 FILEB FILEDD,DD,BUFFD1,,201,1;20
230 003723 003716400000 010
231 003724 DWD VFD 18/FILCCD;1/1;17/0
232 000060 BURF 35S 202 BUFFER FOR FILEDD
232 JSE
BASIC SA COURSE - GEFRC EXAMPLES

PROCESSING FIXED LENGTH RECORDS

235 *
236 *
237 * THIS IS AN EXAMPLE OF CODING TO ESTABLISH
238 * RECORD SIZE OF INPUT AND OUTPUT FILES
239 * AND THE TRANSFER OF DATA TO AND FROM
240 * WORKING STORAGE AREA WSC
241 *
242 *
243 CALL OPEN(CWD,1)

000060 040000701600 030
000061 0000671000 010
000062 007210000363 010
000063 003513000000 010
000064 000001000000 000
000065 040000701800 030
000066 000072100000 010
000067 007210000364 010
244 CALL OPEN(DWD,1)

000070 003723000000 010
000071 000001000000 000
000072 000024 2250 03 000
000073 0000077960 000
000074 010000704000 030
000075 000102710000 010
245 LDX5 #20,DU PLACE RECORD SIZE OF INPUT FILE IN FGB

000076 007210000367 010
000077 003500000000 010
000100 000110000000 010
000101 003620000000 010
000102 000024 2250 03 000
000103 000071 7750 00 010
246 STX5 FILECC+1

000104 020000701000 030
000105 000111710000 010
000106 007210000372 010
000107 003716000000 010
000110 003662000000 010
000111 070000700000 010
000112 000116710000 010
247 CALL SET(FILECC,ENDC,WSC)

000113 007210000373 010
000114 003513000000 010
000115 000000100000 000
000116 070000701400 030
000117 000123710000 010
248 LDX5 #20,DU PLACE RECORD SIZE OF OUTPUT FILE IN FCB

000120 007210000374 010
000121 003723000000 010
000122 000001000000 000
249 STX5 FILEDD+1
250 CALL PJT(FILEDD,WSC)
251 ENDC CALL CLOSE(CWD,1)
252 CALL CLOSE(DWD,1)
CALL ATREC EXAMPLES

255 *
256 *
257 * THE CALL ATREC CAN BE USED TO CREATE A
258 * LOGICAL RECORD AND DIRECT IT TO ANY OUTPUT
259 * MEDIA DEPENDING ON THE DEVICE SPECIFIED
260 * ON THE FILE CONTROL CARD
261 *
262 * FILE CONTROL BLOCK, FILE DESIGNATOR WORD,
263 * AND BUFFER FOR OUTPUT FILE OUTFL
264 *
265 *
266 004236 53E COUNT
267 004236 51BC OUTFL, EE, OTBUFF
268 *
269 004235 004236400000 010
270 OTWD VPD 18/OUTFL, 1/1/70
271 OTBUFF ASS 321 BUFFER FOR OUTFL
272 000123 USE
CALL WTREC EXAMPLES

275 *
276 *
277 * THIS IS AN EXAMPLE USING CALL WTREC
278 * TO CREATE A LOGICAL RECORD AND DIRECT
279 * IT TO TAPE OR DISK DEPENDING ON THE
280 * FILE CONTROL CARD
281 *
282 *
283 _DA #30,DU
284 STA OUTFL='1' FILE CONTROL BLOCK
285 CALL WTREC(OUTFL,WSEET,=5,=20)

286 *
287 *
288 * A LOGICAL RECORD WILL BE CREATED:
289 * USING THE CONTENTS OF THIS
290 * FIELD
291 *
292 *

293 JSE COUNT
294 WSEET JSE
295 JSE
### BASIC SA COURSE - GEFRC EXAMPLES

#### CALL: WREC EXAMPLES

```plaintext
298 *
299 *
300 * THE FOLLOWING EXAMPLES CREATE A LOGICAL RECORD
301 * USING THE CALL WREC AND DIRECT THE RECORD
302 * TO THE PRINTER OR CARD PUNCH THROUGH THE
303 * USE OF A SYSOUT EE FILE CONTROL CARD
304 *
305 * THIS IS AN EXAMPLE USING CALL WREC
306 * TO PUNCH A BINARY CARD
307 *
308 *
309      LOAD #27,DU      PLACE THE RECORD SIZE IN
310      STA OUTFL,1     FILE CONTROL BLOCK
311 CALL WREC(OUTFL,JTSTR,=1,REP2)  
312 *
313 *
314      JTSTR #3I     JSE COUNT
315 #3I 9, THIS INFORMATION
316 *
317 #3I 9, WILL BE PUNCHED
318 *
319 #3I 9, IN BINARY
```

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>000134</td>
<td>000033</td>
</tr>
<tr>
<td>000135</td>
<td>004247</td>
</tr>
<tr>
<td>000136</td>
<td>015020</td>
</tr>
<tr>
<td>000137</td>
<td>001400</td>
</tr>
<tr>
<td>000140</td>
<td>007210</td>
</tr>
<tr>
<td>000141</td>
<td>004240</td>
</tr>
<tr>
<td>000142</td>
<td>009013</td>
</tr>
<tr>
<td>000143</td>
<td>007240</td>
</tr>
<tr>
<td>000144</td>
<td>005040</td>
</tr>
<tr>
<td>005013</td>
<td>002063</td>
</tr>
<tr>
<td>005014</td>
<td>003145</td>
</tr>
<tr>
<td>005015</td>
<td>004216</td>
</tr>
<tr>
<td>005016</td>
<td>002020</td>
</tr>
<tr>
<td>005017</td>
<td>002020</td>
</tr>
<tr>
<td>005020</td>
<td>002020</td>
</tr>
<tr>
<td>005021</td>
<td>002020</td>
</tr>
<tr>
<td>005022</td>
<td>002020</td>
</tr>
<tr>
<td>005023</td>
<td>002020</td>
</tr>
<tr>
<td>005024</td>
<td>002020</td>
</tr>
<tr>
<td>005025</td>
<td>002222</td>
</tr>
<tr>
<td>005026</td>
<td>004523</td>
</tr>
<tr>
<td>005027</td>
<td>002020</td>
</tr>
<tr>
<td>005030</td>
<td>002020</td>
</tr>
<tr>
<td>005031</td>
<td>002020</td>
</tr>
<tr>
<td>005032</td>
<td>002020</td>
</tr>
<tr>
<td>005033</td>
<td>002020</td>
</tr>
<tr>
<td>005034</td>
<td>002020</td>
</tr>
<tr>
<td>005035</td>
<td>002020</td>
</tr>
<tr>
<td>005036</td>
<td>003145</td>
</tr>
<tr>
<td>005037</td>
<td>002020</td>
</tr>
<tr>
<td>005040</td>
<td>002020</td>
</tr>
<tr>
<td>005041</td>
<td>002020</td>
</tr>
<tr>
<td>005042</td>
<td>002020</td>
</tr>
<tr>
<td>005043</td>
<td>002020</td>
</tr>
<tr>
<td>005044</td>
<td>002020</td>
</tr>
<tr>
<td>005045</td>
<td>002020</td>
</tr>
</tbody>
</table>
```
CALL XTRC EXAMPLES

318 *
319 *
320 *
321 *  SYSOUT WILL SORT TOGETHER ALL RECORDS HAVING THE SAME REPORT CODE
322 *
323 *
324 RE=22 JCT 000000000022 REPORT CODE OF 22
325 USE 000145
CALL WTREC EXAMPLES

328  *
329  *
330  * THIS IS AN EXAMPLE USING CALL WTREC
331  * TO PUNCH A HOLLERITH CARD
332  *
333  *
334  LDA  #14,DU
335  STA  OUTFL+1
336  CALL WTREC(OUTFL, SPUN, =2, REP22)
337  *
338  *
339  JSE  COUNT
340  HSPUN  ECI  9, THIS INFORMATION WILL BE
341  ECI  5, PUNCHED IN HOLLERITH
000145 000016 2380 03 000
000146 004247 7950 00 010
000147 150007093000 030
000150 00156710000 010
000151 007210009320 010
000152 004246009320 010
000153 005047009320 010
000154 007215009320 010
000155 009046009320 010
005047 202063303162 000
005050 203142266451 000
005051 442165311645 000
005052 206631435320 000
005053 222220204020 000
005054 202020204020 000
005055 202020204020 000
005056 202020204020 000
005057 202020204020 000
005060 202047649323 000
005061 302652403145 000
005062 203046433325 000
005063 51363306020 000
005064 202020204020 000
000156 342  JSE
CALL WTREC EXAMPLES

345
346
347 THIS IS AN EXAMPLE USING CALL WTREC
348 TO PRINT A LINE
349
350
351 LDA #21, D1
352 STA OUTFIL +1
353 MOVE RECORD SIZE INCLUDING
354 THE SLEW CHARACTER TO THE FCB
355 CALL WTREC (OUTFL, PRWS, PRLIN, REP22)

356 JSE COUNT
357 PRWS 3CI 9, THIS INFORMATION

358 3CI 9, WILL BE PRINTED

359 3CI 2,
360 OCT 770400000000 PRINTER SLEW CHARACTER
361
362
363 A MEDIA CODE OF 3 DIRECTS THIS
364 RECORD THROUGH SYSOJUT TO THE PRINTER
365
366
367 PRLIN ZERO 0,3 MEDIA CODE
05112 000000 000003 000
CALL PRINT EXAMPLES

370 *
371 *
372 * THIS IS AN EXAMPLE USING CALL PRINT
373 * TO WRITE A PRINT LINE
374 *
375 *
005113
005130 005123400000 010
005131
000167

376 FILCB FLFF, FF, BJFF
377 FFWD VFD 18/FLFF,1/1,17/0
378 BUFF: BS 321
379 JSE
380 *
381 *
382 LDX6 @22, DU
383 STX6 FLFF*1
384 CALL PRINT(FLFF,FFWS*2)

385 *
386 *
387 JSE COUNT
388 FFWS 3CI 9, THIS INFORMATION

389 3CI 9, WILL BE PRINTED

390 3CI 4,
CALL PUNCH EXAMPLES

394 *
395 *
396 * THIS IS AN EXAMPLE USING CALL PUNCH
397 * TO PUNCH BINARY CARDS
398 *
399 *
400 CALL PUNCH('FLFF', 'BINWS', '0')

000177 120000701000 030 000200 000205710000 010
000201 007210090200 010 000202 005123009000 010
000203 005660009000 010 000204 072216100000 010
005630

005660 202063003162 000 005661 203149232194 000
005662 442163132664 000 005663 202020202020 000
005664 202020202020 000 005665 202020202020 000
005666 202020202020 000 005667 202020202020 000
005668 202020202020 000 005669 202020202020 000
005670 202020202020 000 005671 202066313343 000
005672 202033235056 000 005673 442230254020 000
005674 202020202020 000 005675 202020202020 000
005676 202020202020 000 005677 202020202020 000
005678 202020202020 000 005700 202020202020 000
005701 202020202020 000 005702 202031452020 000
005703 314521517020 000 005704 202020202020 000
005705 202020202020 000 005706 202020202020 000
005707 202020202020 000

401 JSE COUNT
402 BINWS 3DJ 9, THIS INFORMATION

403 3DJ 9, WILL BE PUNCHED

404 3DJ 6, IN BINARY

405 JSE
CALL: PUNCH EXAMPLES

408
409
410 This is an example using CALL PUNCH
411 TO PUNCH HOLLERITH CARDS
412
413
414 CALL PUNCH(FLFF,PJNWS,=1)

000205  120000701000  030
000206  000213710000  010
000207  007210009636  010
000210  005123000000  010
000211  005710000000  010
000212  007214000300  010

005710
005711  202023303162  000
005713  205614513200  000
005714  223520471445  000
005715  230023246220  000
005716  2020202020  000
005717  2020202020  000
005718  2020202020  000
005719  2020202020  000
005720  2020202020  000
005721  2020202020  000
005722  46434325431  000
005723  6330202020  000

415
416
417 JSE COUNT
418 PUNWS 301 9, THIS INFORMATION WILL BE PUNCHED

419 301 3, IN HOLLERITH

000213  420 JSE
CALL IOEDIT

423 *
424 *
425 * THE CALL IOEDIT Initializes parameters
426 * that will be used as editing functions.
427 * When either call PRINT or CALL PUNCH
428 * are used
429 *
430 *
431 * CALL IOEDIT(LLIST,5)

000213 160000701000 030
000214 000220710000 010
000215 007210000657 010
000216 005724000000 010
000217 000000500000 000

LIST OF CONTROL PARAMETERS AND CODES

432 *
433 *
434 *
435 *
436 *
437 009724
438 * JSEP COUNT
439 * LLIST ZERO HEAD1,1
440 * ZERO HEAD2,2
441 * ZERO PCHCD,3
442 * ZERO RPCHD,7
443 *
444 *
445 * TEN WORDS TO BE INSERTED IN PRINT POSITIONS
446 * 91-131 = 90 of the first line of the heading
447 *
448 * WITH CALL PRINT
449 *
450 005724
451 202063330162 000
452 203145269651 000
453 203145269645 000
454 206631453220 000
455 202031453220 000
456 202031453220 000
457 *
458 * HEAD1: 3CI 5, THIS INFORMATION WILL BE
459 *
460 * 3CI 5, INSERTED IN FIRST LINE
461 *
462 *
463 *
464 * TWENTY WORDS TO BE USED AS THE SECOND
465 *
466 * HEADING LINE OF EACH PAGE WITH CALL PRINT
467 *
468 *
469 *
470 *
471 *
472 *
473 *
474 *
475 * HEAD2: 3CI 9, THIS INFORMATION WILL
CALL: IEDIT

005746 206631434320 000
005747 202020202020 000
005750 202020202020 000
005751 202020202020 000
005752 202020202020 000
005753 202020202020 000
005754 202022220220 000
005755 202145345224 000
005756 2021220330 000
005757 202020202020 000
005760 202020202020 000
005761 202020202020 000
005762 202020202020 000
005763 202020202020 000
005764 202020202020 000
005765 202026262220 000
005766 302221243145 000

459. 3CI 9. BE PRINTED AS THE

460 3CI 2. SUB HEADING

227

005767 272151000001 000
005770 000020206020 000

469 PCHC5 3CI 2, GAR00103

470

471

472 THIS IS USED TO CHANGE THE REPORT CODE WITH THE CALL PRINT

473 374 TOTAL 74 IS ASSUMED BY SYSOUT

475

476

005771 000000000024 000

477 RCD DEC 20

478

479

480 THE PAGES OF THE OUTPUT REPORT WILL

481 BE NUMBERED BEGINNING WITH THIS VALUE

482

483

005772 00000010000 000

484 RCHR 3CI 1,000100
FLOATING LOAD/STORE OPERATIONS

Y-PAIR

Y

0 78 35 36 71

EXP. FRACTIONAL MANTISSA

0 27 35 63 71

A-REGISTER Q-REGISTER

ZEROS S. P. D.P.

0 1 2 3 4 5 6 7

RANGE +127 TO -128

EXPONENT REGISTER
FLOATING LOAD/STORE

FLD  FST
DFLD  DFST
LDE  STE
FSTR
LOADING EXPONENT REGISTER

EXAMPLE: \[ \text{LDE} = 35B25, \text{DU} \]
FLOATING POINT ARITHMETIC

FAD
DFAD
ADE
UFA
DUFA
FNO

FSB
DFSB
UFS
DUFS
FNEG
FLOATING POINT

MULTIPLY AND DIVIDE

FMP       FDV
DFMP      DFDV
UFM       FDI
DUFM      DFDI
FLOATING POINT COMPARE

FCMP   Y
DFCMP  Y-PAIR
FCMG   Y
DFCMG  Y-PAIR
<table>
<thead>
<tr>
<th>ASSEMBLY NOTATION</th>
<th>S</th>
<th>EXPONENT</th>
<th>S MANTISSA</th>
<th>OCTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>= .1</td>
<td>1</td>
<td>11111101</td>
<td>0110 011 001 100 110 011 001 100 110</td>
<td>772631463146</td>
</tr>
<tr>
<td>= .2</td>
<td>1</td>
<td>11111110</td>
<td>0110 011 001 100 110 011 001 100 110</td>
<td>774631463146</td>
</tr>
<tr>
<td>= .3</td>
<td>1</td>
<td>11111111</td>
<td>0100 110 011 001 100 110 011 001 101</td>
<td>77643146315</td>
</tr>
<tr>
<td>= .4</td>
<td>1</td>
<td>11111111</td>
<td>0110 011 001 100 110 011 001 100 110</td>
<td>776631463146</td>
</tr>
<tr>
<td>= .5</td>
<td>0</td>
<td>00000000</td>
<td>0100 000 000 000 000 000 000 000 000</td>
<td>000400000000</td>
</tr>
<tr>
<td>= .6</td>
<td>0</td>
<td>00000000</td>
<td>0100 110 011 001 100 000 011 001 101</td>
<td>000463146315</td>
</tr>
<tr>
<td>= .7</td>
<td>0</td>
<td>00000000</td>
<td>0101 100 110 011 001 100 110 011 010</td>
<td>00054314632</td>
</tr>
<tr>
<td>= .8</td>
<td>0</td>
<td>00000000</td>
<td>0110 011 001 100 110 011 001 100 110</td>
<td>000631463146</td>
</tr>
<tr>
<td>= .9</td>
<td>0</td>
<td>00000000</td>
<td>0111 001 100 110 011 001 100 110 011</td>
<td>000714631463</td>
</tr>
<tr>
<td>= .10</td>
<td>1</td>
<td>11111101</td>
<td>0110 011 001 100 110 011 001 100 110</td>
<td>772631463146</td>
</tr>
<tr>
<td>= 0.</td>
<td>1</td>
<td>00000000</td>
<td>0000 000 000 000 000 000 000 000 000</td>
<td>400000000000</td>
</tr>
<tr>
<td>= 1.</td>
<td>0</td>
<td>00000001</td>
<td>0100 000 000 000 000 000 000 000 000</td>
<td>002400000000</td>
</tr>
<tr>
<td>= 2.</td>
<td>0</td>
<td>00000010</td>
<td>0100 000 000 000 000 000 000 000 000</td>
<td>004400000000</td>
</tr>
<tr>
<td>= 3.</td>
<td>0</td>
<td>00000010</td>
<td>0110 000 000 000 000 000 000 000 000</td>
<td>004600000000</td>
</tr>
<tr>
<td>= 4.</td>
<td>0</td>
<td>00000110</td>
<td>0100 000 000 000 000 000 000 000 000</td>
<td>006400000000</td>
</tr>
<tr>
<td>= 5.</td>
<td>0</td>
<td>00000110</td>
<td>0101 000 000 000 000 000 000 000 000</td>
<td>006500000000</td>
</tr>
<tr>
<td>ASSEMBLY NOTATION</td>
<td>FLOATING POINT MEMORY WORD</td>
<td>OCTAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(-1.)</td>
<td>[0 1 0 0 0 0 0 0 0 0]</td>
<td>[1 0 0 0 0 0 0 0 0 0 0 0]</td>
<td>[00100000000]</td>
<td></td>
</tr>
<tr>
<td>(-2.)</td>
<td>[0 1 0 0 0 0 0 0 0 1]</td>
<td>[1 0 0 0 0 0 0 0 0 0 0 0]</td>
<td>[00300000000]</td>
<td></td>
</tr>
<tr>
<td>(-3.)</td>
<td>[0 1 0 0 0 0 0 0 1 0]</td>
<td>[1 0 1 0 0 0 0 0 0 0 0 0]</td>
<td>[00520000000]</td>
<td></td>
</tr>
<tr>
<td>(-4.)</td>
<td>[0 1 0 0 0 0 0 0 1 0]</td>
<td>[1 0 0 0 0 0 0 0 0 0 0 0]</td>
<td>[00500000000]</td>
<td></td>
</tr>
<tr>
<td>(-5.)</td>
<td>[0 1 0 0 0 0 0 1 1]</td>
<td>[1 0 1 1 0 0 0 0 0 0 0 0]</td>
<td>[00730000000]</td>
<td></td>
</tr>
<tr>
<td>(-.1)</td>
<td>[1 1 1 1 1 1 0 1]</td>
<td>[1 0 0 1 1 0 0 0 1 1 0 0]</td>
<td>[773146314632]</td>
<td></td>
</tr>
<tr>
<td>(-.2)</td>
<td>[1 1 1 1 1 1 0]</td>
<td>[1 0 0 1 1 0 0 1 1 0 1 0]</td>
<td>[775146314632]</td>
<td></td>
</tr>
<tr>
<td>(-.3)</td>
<td>[1 1 1 1 1 1]</td>
<td>[1 0 1 1 0 1 1 0 1 0 1 1]</td>
<td>[777146314632]</td>
<td></td>
</tr>
<tr>
<td>(-.4)</td>
<td>[1 1 1 1 1 1]</td>
<td>[1 0 0 1 1 0 1 1 0 1 1 0]</td>
<td>[777146314632]</td>
<td></td>
</tr>
<tr>
<td>(-.5)</td>
<td>[1 1 1 1 1 1]</td>
<td>[1 1 0 0 0 0 0 0 0 0 0 0]</td>
<td>[77700000000]</td>
<td></td>
</tr>
</tbody>
</table>
# Floating Point

<table>
<thead>
<tr>
<th>ASSEMBLY NOTATION</th>
<th>FLOATING POINT MEMORY WORD</th>
<th>OCTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.55E1</td>
<td>0 0 0 0 0 1 1 0 0 0 0 1 1 1 0 0</td>
<td>014174000000</td>
</tr>
<tr>
<td>1.55E-1</td>
<td>1 1 1 1 1 1 1 0 0 0 1 0 1 1 0 0 0 0</td>
<td>010760000000</td>
</tr>
<tr>
<td>5.12E2</td>
<td>0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0</td>
<td>024400000000</td>
</tr>
<tr>
<td>5.2E-1</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>000000000000</td>
</tr>
<tr>
<td>12345.12345</td>
<td>0 0 0 0 1 1 0 0 0 1 1 1 0 1 0 0 0 0</td>
<td>034601621763</td>
</tr>
<tr>
<td>320.12345</td>
<td>0 0 0 0 0 1 0 0 0 1 0 1 0 0 0</td>
<td>TRUNCATED @ BIT 17</td>
</tr>
</tbody>
</table>
LDA 3000,3*  C(X3)=50

3000  LDA  3*

TEMP. EFF. ADDR. = 3000+C(X3)=3050

(3050) XXX 4000

4000  XXX

RI MODIFICATION
LDQ

4000,0* C(X0)=100

4000 | LDQ | 0*

(4100) XXX 4200,1* C(X1)=200

4200 | XXX | 1*

(4400) XXX 4600,2* C(X2)=300

4600 | XXX | 2*

(4900) XXX 5000,1 C(X1)=(STILL)200

5000 | XXX | 1

RI MODIFICATION
REGISTER INDIRECT (RI)

PERFORM THE REGISTER MODIFICATION TO OBTAIN THE ADDRESS OF THE INDIRECT WORD.

IF THE INDIRECT WORD SPECIFIES FURTHER INDIRECTION, THEN FETCH THE NEXT INDIRECT WORD.

CEASE INDIRECTION WHEN THE CANDIDATE FOR THE ACTUAL EFFECTIVE ADDRESS IS EITHER:

1. NOT INDEXED AT ALL
2. INDEXED BY A REGISTER (R) TYPE
3. INDEXED BY AN INDIRECT THEN TALLY (IT) TYPE

THE VARIATIONS DU AND DL OF REGISTER (R) MODIFICATION CANNOT BE USED WITH (RI) MODIFICATION.

WHEN WORKING WITH (RI) WE USE THE:

- INDEX REGISTERS
- A-REGISTER
- Q-REGISTER
- INSTRUCTION COUNTER

\(X_n (n = 0-7)\)

\(\text{UPPER AND LOWER PORTIONS}\)

\(\text{UPPER AND LOWER PORTIONS}\)

\(IC\)
### ALGORITHMS:

**REGISTER AND INDIRECT (RI)**

<table>
<thead>
<tr>
<th>PERMISSIBLE REGISTERS</th>
<th>EFFECTIVE ADDRESS</th>
<th>CODING FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xn</td>
<td>$y_i$</td>
<td>$y + C(Xn) \ 0-17$</td>
</tr>
<tr>
<td>AU</td>
<td>$y_i$</td>
<td>$y + C(A) \ 0-17$</td>
</tr>
<tr>
<td>AL</td>
<td>$y_i$</td>
<td>$y + C(A) \ 18-35$</td>
</tr>
<tr>
<td>QU</td>
<td>$y_i$</td>
<td>$y + C(Q) \ 0-17$</td>
</tr>
<tr>
<td>QL</td>
<td>$y_i$</td>
<td>$y + C(Q) \ 18-35$</td>
</tr>
<tr>
<td>IC</td>
<td>$y_i$</td>
<td>$y + C(IC) \ 0-17$</td>
</tr>
<tr>
<td>N</td>
<td>$y_i$</td>
<td>$y$</td>
</tr>
</tbody>
</table>
INDIRECT THEN REGISTER

IR

STORE REGISTER DESIGNATOR

FETCH CONTENTS INDIR WORD

R OR IT

YES

FETCH SAVED TD

STOP

NO

RI

NO

YES

ADD REGISTER

TM = IR
LDA 3000,*5 X5 SAVED

(3000) XXX 1000,*7 X5 NOW REPLACED BY X7

(1000) XXX 200 CEASE INDIrection AND MODIFY USING X7

IR MODIFICATION
INDIRECT REGISTER (IR)

REGISTER SPECIFIED IS PLACED IN A SAVE-STORE AREA.

ADDRESS IS USED TO LOCATE THE INDIRECT WORD.

PROCESS IS CONTINUED UNTIL THE CANDIDATE FOR THE ACTUAL EFFECTIVE ADDRESS IS EITHER:
   1. NOT INDEXED AT ALL
   2. INDEXED BY A REGISTER (R) TYPE
   3. INDEXED BY AN INDIRECT THEN TALLY (IT) TYPE

THE VARIATIONS DU AND DL OF REGISTER (R) MODIFICATION WITH (IR) MODIFICATION.

WHEN THE SEARCH FOR THE ACTUAL EFFECTIVE ADDRESS IS TERMINATED, THE LAST MODIFICATION PERFORMED IS DEPENDENT UPON THE CONTENTS OF THE SAVE-STORE AREA.

WHEN WORKING WITH (IR) WE USE THE:

INDEX REGISTER
A-REGISTER
Q-REGISTER
INSTRUCTION COUNTER

Xn (n = 0 - 7)
(UPPER AND LOWER PORTIONS)
(UPPER AND LOWER PORTIONS)
IC
### ALGORITHMS:

**INDIRECT THEN REGISTER (IR)**

<table>
<thead>
<tr>
<th>PERMISSIBLE REGISTERS</th>
<th>EFFECTIVE ADDRESS</th>
<th>CODING FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xn</td>
<td>( y_i = y ) (SAVE C(Xn)0-17)</td>
<td>Y,*0-7</td>
</tr>
<tr>
<td>AU</td>
<td>( y_i = y ) (SAVE C(AU)0-17)</td>
<td>Y,*AU</td>
</tr>
<tr>
<td>AL</td>
<td>( y_i = y ) (SAVE C(AL)18-35)</td>
<td>Y,*AL</td>
</tr>
<tr>
<td>QU</td>
<td>( y_i = y ) (SAVE C(QU)0-17)</td>
<td>Y,*QU</td>
</tr>
<tr>
<td>QL</td>
<td>( y_i = y ) (SAVE C(QL)18-35)</td>
<td>Y,*QL</td>
</tr>
<tr>
<td>IC</td>
<td>( y_i = y ) (SAVE C(IC)0-17)</td>
<td>Y,*IC</td>
</tr>
<tr>
<td>N</td>
<td>( y_i = y ) (SAVE NO REGISTER)</td>
<td>Y,*N</td>
</tr>
</tbody>
</table>
LDA 500, *IC  IC SAVED

(500) XXX 300, *2  IC IS REPLACED BY X2

(300) XXX 2000, 3* C(X3)=1000. USE X3; X2 STILL SAVED.

(3000) XXX 200, 4 USE X2; X4 IGNORERED.

IR AND RI MODIFICATION
EXAMPLE 1

FIRST
SECOND + C(A)_0-17
LDA
STA
SECOND, AU*
THIRD, I

THIRD + C(XI)_0-17

EXAMPLE 2

NONE
ONCE
MPY
ARG
ONCE, * TWICE, QU

TWICE + C(QU)

EXAMPLE 3

A
LDA
ARG
A, *QL

B
B + C(Q)_18-35
EXAMPLE 4

B
C
D

LDA
ARG
ARG

B, *3
C, *5
D, *QU
E, 7

E + C(Q)(0 - 17)

EXAMPLE 5

A
B + C(X5)

LDA
ARG
ARG

A, *3
B, 5*
C, 1C

C + C(X3)

EXAMPLE 6

Z

LDA
STA

Z, *3
B, DU*

NOT VALID
EXAMPLE 7

A + C (X3)    LDA    A, 3*
B             STA    B, *3
C             LDQ    C, *N
D + C (X1)    STQ    D, 1*
E             ARG    E, *

F

EXAMPLE 8

START
A + C (X7)    LDA    A, 7*
B + C (X4)    ARG    B, 4*
C + C (X5)    ARG    C, 5*

1    D    ARG    E
2    D    ARG    E, 3
3    D    ARG    E, 1D

E    TALLY    F, 100

1    E
2    E + C (X3)
3    F
FIXED-POINT DATA FORMATS

0
SINGLE PRECISION 35
36-BIT OPERAND

0
DOUBLE PRECISION 35
FIRST 36-BIT OPERAND

36
SECOND 36-BIT OPERAND

HALF WORD
0 17 18
UPPER 18-BIT 35
HALF WORD
LOWER 18-BIT
HALF WORD
FLOATING-POINT DATA FORMATS

SINGLE PRECISION

0 7 8 35
8-BIT EXPONENT 28-BIT MANTISSA

DOUBLE PRECISION

0 7 8 35
8-BIT EXPONENTS FIRST 28 BITS OF MANTISSA

36 71
LAST 36 BITS OF MANTISSA
ALPHANUMERIC DATA FORMATS

6-BIT FORMAT

<table>
<thead>
<tr>
<th>0</th>
<th>5</th>
<th>6</th>
<th>11</th>
<th>12</th>
<th>17</th>
<th>18</th>
<th>23</th>
<th>24</th>
<th>29</th>
<th>30</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHARACTER</td>
<td>CHARACTER</td>
<td>CHARACTER</td>
<td>CHARACTER</td>
<td>CHARACTER</td>
<td>CHARACTER</td>
<td>CHARACTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

9-BIT FORMAT

<table>
<thead>
<tr>
<th>0</th>
<th>8</th>
<th>9</th>
<th>17</th>
<th>18</th>
<th>26</th>
<th>27</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHARACTER</td>
<td>CHARACTER</td>
<td>CHARACTER</td>
<td>CHARACTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DECIMAL FORMATS FOR EIS

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>4</th>
<th>5</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>13</th>
<th>14</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>22</th>
<th>23</th>
<th>26</th>
<th>27</th>
<th>28</th>
<th>31</th>
<th>32</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Z</td>
<td>0</td>
<td>Z</td>
<td>2</td>
<td>3</td>
<td>Z</td>
<td>4</td>
<td>5</td>
<td>Z</td>
<td>6</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PACKED DECIMAL (4-BIT)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>26</th>
<th>27</th>
<th>28</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Z</td>
<td>0</td>
<td>Z</td>
<td>1</td>
<td>Z</td>
<td>2</td>
<td>Z</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ASCII (9-BIT)
### Floating-Point Decimal Formats

#### 9-Bit Characters
- Sign
- \(10^N\)
- \(10^1\)
- \(10^0\)
- \(Z\)
- EXPONENT

#### 4-Bit Characters
- \(Z\) Sign
- \(10^N\)
- \(Z\) \(10^3\)
- \(Z\) \(10^2\)
- \(Z\) \(10^1\)
- \(Z\) \(10^0\)
- \(Z\) EXPONENT
- EVEN (ODD) CHARACTER BOUNDARY, \(N\) ODD (EVEN)

#### 4-Bit Characters
- \(Z\) Sign
- \(10^N\)
- \(10^3\)
- \(10^2\)
- \(10^1\)
- \(10^0\)
- \(Z\) EXPONENT
- ODD (EVEN) CHARACTER BOUNDARY, \(N\) ODD (EVEN)
### Decimal Sign Position Formats

<table>
<thead>
<tr>
<th></th>
<th>PLUS</th>
<th>MINUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-BIT</td>
<td>13\textsubscript{8}</td>
<td>15\textsubscript{8}</td>
</tr>
<tr>
<td>9-BIT</td>
<td>53\textsubscript{8}</td>
<td>55\textsubscript{8}</td>
</tr>
<tr>
<td></td>
<td>000(1)</td>
<td>040(1)</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>006</td>
<td></td>
<td></td>
</tr>
<tr>
<td>007</td>
<td></td>
<td></td>
</tr>
<tr>
<td>010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td></td>
<td></td>
</tr>
<tr>
<td>012</td>
<td></td>
<td></td>
</tr>
<tr>
<td>013</td>
<td></td>
<td></td>
</tr>
<tr>
<td>014</td>
<td></td>
<td></td>
</tr>
<tr>
<td>015</td>
<td></td>
<td></td>
</tr>
<tr>
<td>015</td>
<td></td>
<td></td>
</tr>
<tr>
<td>017</td>
<td></td>
<td></td>
</tr>
<tr>
<td>020</td>
<td>MVE</td>
<td>CSL</td>
</tr>
<tr>
<td>021</td>
<td></td>
<td>CSR</td>
</tr>
<tr>
<td>022</td>
<td></td>
<td></td>
</tr>
<tr>
<td>023</td>
<td></td>
<td></td>
</tr>
<tr>
<td>024</td>
<td></td>
<td></td>
</tr>
<tr>
<td>025</td>
<td>MVNE</td>
<td>S2TL</td>
</tr>
<tr>
<td>026</td>
<td></td>
<td></td>
</tr>
<tr>
<td>027</td>
<td></td>
<td></td>
</tr>
<tr>
<td>030</td>
<td></td>
<td></td>
</tr>
<tr>
<td>031</td>
<td></td>
<td></td>
</tr>
<tr>
<td>032</td>
<td></td>
<td></td>
</tr>
<tr>
<td>033</td>
<td></td>
<td></td>
</tr>
<tr>
<td>034</td>
<td></td>
<td></td>
</tr>
<tr>
<td>035</td>
<td></td>
<td></td>
</tr>
<tr>
<td>036</td>
<td></td>
<td></td>
</tr>
<tr>
<td>037</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1 Octal representation of the operation code consists of three octal digits corresponding to instruction bit positions 18-26 and a zero (0) or one (1) for instruction bit position 27 (the EIS bit).

Figure 1-2. Operation Code Map for the Extended Instructions
|        | 000 | 001 | 002 | 003 | 004 | 005 | 006 | 007 | 010 | 011 | 012 | 013 | 014 | 015 | 016 | 017 | 020 | 021 | 022 | 023 | 024 | 025 | 026 | 027 | 030 | 031 | 032 | 033 | 034 | 035 | 036 | 037 |
|--------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 400(1) |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 440(1) |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 500(1) | A9BD| A8BD| A6BD| A4BD| ABD |      |      |      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 540(1) | ARA0| ARA1| ARA2| ARA3| ARA4| ARA5| ARA6| ARA7| TRIN| TRIF| TMO2| TPNZ| TIN |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 600(1) |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 640(1) | ARN0| ARN1| ARN2| ARN3| ARN4| ARN5| ARN6| ARN7|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 700(1) | SAR0| SAR1| SAR2| SAR3| SAR4| SAR5| SAR6| SAR7|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 740(1) |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

1 Octal representation of the operation code consists of three octal digits corresponding to instruction bit positions 18-26 and a zero(0) or one (1) for instruction bit position 27 (the EIS bit).

Figure 1-2 (cont). Operation Code Map for the Extended Instructions
EIS MULTI-WORD INSTRUCTIONS

<table>
<thead>
<tr>
<th>VARIABLE FIELD</th>
<th>OP CODE</th>
<th>MF 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1ST OPERAND_DESCRIPTOR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2ND OPERAND_DESCRIPTOR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3RD OPERAND_DESCRIPTOR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### SINGLE WORD INSTRUCTIONS

<table>
<thead>
<tr>
<th>Special Format</th>
<th>0</th>
<th>2</th>
<th>3</th>
<th>17</th>
<th>18</th>
<th>27</th>
<th>29</th>
<th>35</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Normal Format</th>
<th>0</th>
<th>17</th>
<th>18</th>
<th>27</th>
<th>29</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operation Code</td>
<td></td>
<td>Information Field</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
OPERAND DESCRIPTORS

BIT STRING

0  3  17  18  19  20  23  24  35
   Y   C   B   N

BIT POSITION
CHARACTER POSITION

ALPHANUMERIC

0  3  17  18  20  21  22  23  24  35
   Y   CN   TA   O   N

DATA TYPE
CHARACTER POSITION

NUMERIC

0  3  17  18  20  21  22  23  24  29  30  35
   Y   CN   TN   S   SF   N

SIGN:
LEADING SIGN, FLOATING POINT
LEADING SIGN, SCALED
TRAILING SIGN, SCALED
NO SIGN

LENGTH
SCALE FACTOR
EIS ADDRESS MODIFICATION

0  2  3  17  18  26  27  28  29  35
| Y | OP | I | TAG | AR | TD |

1  1  1  4
| AR | RL | ID | REG |

NUMBER OF BITS
SYMBOL

AR  ADDRESS REGISTER SPECIFIER
RL  REGISTER OR LENGTH
ID  INDIRECT OPERAND DESCRIPTOR
REG ADDRESS MODIFICATION REGISTER
INDIRECT WORD

<table>
<thead>
<tr>
<th>0</th>
<th>2</th>
<th>3</th>
<th>29</th>
<th>30</th>
<th>31</th>
<th>32</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRESS</td>
<td>AR</td>
<td>00</td>
<td>REG</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ADDRESS REGISTER NUMBER
(IF AR MODIFICATION SPECIFIED)

REGISTER MODIFICATION SPECIFIER
ADDRESS REGISTER MODIFICATION SPECIFIER

AR AND REG FIELDS IDENTICAL IN FUNCTION WITH CORRESPONDING MODIFICATION FIELDS IN INSTRUCTION.
ADDRESS REGISTERS

ADDRESS FIELD

0 TO 7

INCREMENT

ADDRESS REGISTER (24 BITS) 0 TO 7

BIT?

CHARACTER?

WORD?
1. Fetch Instruction

2. 

3. AR of Indirect Word = 1?

4. Mod. Y of Indirect Word w/ AR

5. REG of Indirect Word Contain Code?

6. Mod. Y of Indirect Word w/ REG

7. Fetch Oper. Descriptor From Store

8. MF$_1$ AR = 1?

9. Mod. Y of Operand Descriptor By AR

10. MF$_1$ REG = Code?

11. Mod. Y of Operand Descriptor with REG

12. Fetch Operand from Store

END
SUBPROGRAM VG. 1 - DRIVER

PREFACE

PROGRAM BREAK 3
COMPAK LENGTH 0
V CEAT BITS 5

PRIMARY SYNDEF ENTRY

START 3

SECONDARY SYNDEF ENTRY

BLOCK LENGTH

SYNDEF

1 CALC
2 EDIT
3 INI
4 LOCAT
5 INALT
6 TOPC
7 TOPIA
10 CLTFLT
11 WRTFLP

-255
SUBPROGRAM NO. 1 - DRIVER

1  TIL
2  SYMDEF
3  START
4  READ
5  SKIP
6  CALL
7  CALL
8  CALL
9  CALL
10  CALL
11  CALL
12  CALL

ERROR LINKAGE

END

30 IS THE NEXT AVAILABLE LOCATION. GMAP VERSION JPKA/030271 JPPF/030271 JPPC/030271
THERE WERE NO WARNING FLAGS IN THE ABOVE ASSEMBLY.
<table>
<thead>
<tr>
<th>OCTAL</th>
<th>SYMBOL</th>
<th>REFERENCES BY ALTER NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CALC</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>EDIT</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>FLOAT</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>INIT</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>INPUT</td>
<td>4</td>
</tr>
<tr>
<td>34</td>
<td>F.LIT.</td>
<td>4  5  6  7  8  9  14  12</td>
</tr>
<tr>
<td>10</td>
<td>CLTPLT</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>READ</td>
<td>4  4  11</td>
</tr>
<tr>
<td>0</td>
<td>START</td>
<td>3  2  3</td>
</tr>
<tr>
<td>6</td>
<td>TCBCL</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>TCBIA</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>LWAFLP</td>
<td>12</td>
</tr>
</tbody>
</table>

** 18180 WORDS OF MEMORY WERE USED BY GMAP FOR THIS ASSEMBLY.**
<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>00310</td>
<td>0030</td>
</tr>
<tr>
<td>00002</td>
<td>00310</td>
<td>0030</td>
</tr>
<tr>
<td>00001</td>
<td>00310</td>
<td>0030</td>
</tr>
<tr>
<td>00001</td>
<td>00310</td>
<td>0030</td>
</tr>
<tr>
<td>00001</td>
<td>00310</td>
<td>0030</td>
</tr>
<tr>
<td>00001</td>
<td>00310</td>
<td>0030</td>
</tr>
<tr>
<td>00001</td>
<td>00310</td>
<td>0030</td>
</tr>
<tr>
<td>00001</td>
<td>00310</td>
<td>0030</td>
</tr>
<tr>
<td>00001</td>
<td>00310</td>
<td>0030</td>
</tr>
<tr>
<td>00001</td>
<td>00310</td>
<td>0030</td>
</tr>
<tr>
<td>00001</td>
<td>00310</td>
<td>0030</td>
</tr>
<tr>
<td>00001</td>
<td>00310</td>
<td>0030</td>
</tr>
<tr>
<td>00001</td>
<td>00310</td>
<td>0030</td>
</tr>
<tr>
<td>00001</td>
<td>00310</td>
<td>0030</td>
</tr>
</tbody>
</table>

**Notes:**
- TL: TIME
- SC: SUBROUTINE
- CL: CALL
- LT: LTR
- MN: START
- CT: GOTO
- L: LOAD
- S: SAVE
- M: CONTROL
- R: RETURM
- A: ADD
- C: CLEAR
- D: DIRECTION
- F: FREQUENCY
- H: HALT
- I: INCREMENT
- P: PRINT
- S: SIZE
- T: TIME
- U: UPDATE
- V: VALUE
- X: X-VALUE
- Y: Y-VALUE
- Z: Z-VALUE

**Conversion:**
- MINUTES TO HOURS
- HOURS TO MINUTES

**Special Instructions:**
- LD: LOAD
- STA: STORE
- ST: STORE
- SBR: SAVE
- RES: RESET
- CLR: CLEAR
- ADD: ADD
- SUB: SUBTRACT
- MUL: MULTIPLY
- DIV: DIVIDE
- STR: STORE
- LTR: LOAD
- DTR: DISPLAY
- CRT: CLEAR
- RTR: RETURN
- JZ: JUMP ZERO
- JN: JUMP NONZERO
- JNZ: JUMP NONZERO
- JNZX: JUMP NONZERO X
- JNZY: JUMP NONZERO Y
- JNZZ: JUMP NONZERO Z
- JNZU: JUMP NONZERO U
- JNZF: JUMP NONZERO F
- JNZS: JUMP NONZERO S
- JNZT: JUMP NONZERO T
- JNZV: JUMP NONZERO V
- JNZX: JUMP NONZERO X
- JNZY: JUMP NONZERO Y
- JNZZ: JUMP NONZERO Z
- JNZU: JUMP NONZERO U
- JNZF: JUMP NONZERO F
- JNZS: JUMP NONZERO S
- JNZT: JUMP NONZERO T
- JNZV: JUMP NONZERO V
- JNZX: JUMP NONZERO X
- JNZY: JUMP NONZERO Y
- JNZZ: JUMP NONZERO Z
- JNZU: JUMP NONZERO U
- JNZF: JUMP NONZERO F
- JNZS: JUMP NONZERO S
- JNZT: JUMP NONZERO T
- JNZV: JUMP NONZERO V
- JNZX: JUMP NONZERO X

**Special Characters:**
- @: AT
- #: NUMBER
- &: AND
- |: OR
- ^: NOT
- £: POUND
- ¥: YEN
- €: EURO
- £: POUND
SUBPROGRAM NO. 2 = IN/OUT/PLUT - BELL INPUT

BETTER TRY FOR INPUT

CALL GET(CRED, ELCHE, FLUHR)

RETURN INPT

CALL PLACE(FUNFD, FUNKT, 21)

RETURN OUTPUT

CALL CKWRK, DEC, 0

CALL CKWRT, 0

CALL CKTIME, 0

MINS = 0

MINS = 1

FUNCTION

MCHEWELL MARKETING EDUCATION MPAR COURSE
BETTER TRY FOR INPUT

335 IS THE NEXT AVAILABLE LOCATION. GMAP VERSION JPPA/C30271 JMPE/U30271 JMPC/030271

THERE WERE NO WARNING FLAGS IN THE ABOVE ASSEMBLY
SUBPROGRAM NO. 3 - DATA MANIPULATION SUBROUTINES

FLOAT SUBROUTINE

PREFACE

PROGRAM BREAK 265
COMMCA LENGTH 0
V COLAT BITS 5

PRIMARY SYMDEF ENTRY

FLOAT 0
CALC 102
TOSIA 146
TOBCC 211
EDIT 223

SECONDARY SYMDEF ENTRY

BLOCK LENGTH
1 ID,ARK 62
SYMREF
SUBPROGRAM NO. 3 - DATA MANIPULATION SUBROUTINES

FLOAT SUBROUTINE

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TTL</td>
</tr>
<tr>
<td>2</td>
<td>TILS</td>
</tr>
<tr>
<td>4</td>
<td>FLOAT</td>
</tr>
<tr>
<td>5</td>
<td>LOAD X6 WITH 4</td>
</tr>
<tr>
<td>6</td>
<td>LOAD X7 WITH 8</td>
</tr>
<tr>
<td>7</td>
<td>LOAD X8 WITH 8</td>
</tr>
<tr>
<td>8</td>
<td>LOAD X9 WITH 8</td>
</tr>
<tr>
<td>9</td>
<td>LOAD X10 WITH 8</td>
</tr>
<tr>
<td>10</td>
<td>LOAD X11 WITH 8</td>
</tr>
<tr>
<td>11</td>
<td>LOAD X12 WITH 8</td>
</tr>
<tr>
<td>12</td>
<td>LOAD X13 WITH 8</td>
</tr>
<tr>
<td>13</td>
<td>LOAD X14 WITH 8</td>
</tr>
<tr>
<td>14</td>
<td>LOAD X15 WITH 8</td>
</tr>
<tr>
<td>15</td>
<td>LOAD X16 WITH 8</td>
</tr>
<tr>
<td>16</td>
<td>LOAD X17 WITH 8</td>
</tr>
<tr>
<td>17</td>
<td>LOAD X18 WITH 8</td>
</tr>
<tr>
<td>18</td>
<td>LOAD X19 WITH 8</td>
</tr>
<tr>
<td>19</td>
<td>LOAD X20 WITH 8</td>
</tr>
<tr>
<td>20</td>
<td>LOAD X21 WITH 8</td>
</tr>
<tr>
<td>21</td>
<td>LOAD X22 WITH 8</td>
</tr>
<tr>
<td>22</td>
<td>LOAD X23 WITH 8</td>
</tr>
<tr>
<td>23</td>
<td>LOAD X24 WITH 8</td>
</tr>
<tr>
<td>24</td>
<td>LOAD X25 WITH 8</td>
</tr>
<tr>
<td>25</td>
<td>LOAD X26 WITH 8</td>
</tr>
<tr>
<td>26</td>
<td>LOAD X27 WITH 8</td>
</tr>
<tr>
<td>27</td>
<td>LOAD X28 WITH 8</td>
</tr>
<tr>
<td>28</td>
<td>LOAD X29 WITH 8</td>
</tr>
<tr>
<td>29</td>
<td>LOAD X30 WITH 8</td>
</tr>
<tr>
<td>30</td>
<td>LOAD X31 WITH 8</td>
</tr>
<tr>
<td>31</td>
<td>LOAD X32 WITH 8</td>
</tr>
<tr>
<td>32</td>
<td>LOAD X33 WITH 8</td>
</tr>
<tr>
<td>33</td>
<td>LOAD X34 WITH 8</td>
</tr>
<tr>
<td>34</td>
<td>LOAD X35 WITH 8</td>
</tr>
<tr>
<td>35</td>
<td>LOAD X36 WITH 8</td>
</tr>
<tr>
<td>36</td>
<td>LOAD X37 WITH 8</td>
</tr>
<tr>
<td>37</td>
<td>LOAD X38 WITH 8</td>
</tr>
<tr>
<td>38</td>
<td>LOAD X39 WITH 8</td>
</tr>
<tr>
<td>39</td>
<td>LOAD X40 WITH 8</td>
</tr>
<tr>
<td>40</td>
<td>LOAD X41 WITH 8</td>
</tr>
<tr>
<td>41</td>
<td>LOAD X42 WITH 8</td>
</tr>
<tr>
<td>42</td>
<td>LOAD X43 WITH 8</td>
</tr>
<tr>
<td>43</td>
<td>LOAD X44 WITH 8</td>
</tr>
<tr>
<td>44</td>
<td>LOAD X45 WITH 8</td>
</tr>
<tr>
<td>45</td>
<td>LOAD X46 WITH 8</td>
</tr>
<tr>
<td>46</td>
<td>LOAD X47 WITH 8</td>
</tr>
<tr>
<td>47</td>
<td>LOAD X48 WITH 8</td>
</tr>
<tr>
<td>48</td>
<td>LOAD X49 WITH 8</td>
</tr>
<tr>
<td>49</td>
<td>LOAD X50 WITH 8</td>
</tr>
<tr>
<td>50</td>
<td>LOAD X51 WITH 8</td>
</tr>
<tr>
<td>51</td>
<td>LOAD X52 WITH 8</td>
</tr>
<tr>
<td>52</td>
<td>LOAD X53 WITH 8</td>
</tr>
<tr>
<td>53</td>
<td>LOAD X54 WITH 8</td>
</tr>
<tr>
<td>54</td>
<td>LOAD X55 WITH 8</td>
</tr>
<tr>
<td>55</td>
<td>LOAD X56 WITH 8</td>
</tr>
<tr>
<td>56</td>
<td>LOAD X57 WITH 8</td>
</tr>
<tr>
<td>57</td>
<td>LOAD X58 WITH 8</td>
</tr>
<tr>
<td>58</td>
<td>LOAD X59 WITH 8</td>
</tr>
<tr>
<td>59</td>
<td>LOAD X60 WITH 8</td>
</tr>
<tr>
<td>60</td>
<td>LOAD X61 WITH 8</td>
</tr>
<tr>
<td>61</td>
<td>LOAD X62 WITH 8</td>
</tr>
<tr>
<td>62</td>
<td>LOAD X63 WITH 8</td>
</tr>
<tr>
<td>63</td>
<td>LOAD X64 WITH 8</td>
</tr>
<tr>
<td>64</td>
<td>LOAD X65 WITH 8</td>
</tr>
<tr>
<td>65</td>
<td>LOAD X66 WITH 8</td>
</tr>
<tr>
<td>66</td>
<td>LOAD X67 WITH 8</td>
</tr>
<tr>
<td>67</td>
<td>LOAD X68 WITH 8</td>
</tr>
<tr>
<td>68</td>
<td>LOAD X69 WITH 8</td>
</tr>
<tr>
<td>69</td>
<td>LOAD X70 WITH 8</td>
</tr>
<tr>
<td>70</td>
<td>LOAD X71 WITH 8</td>
</tr>
<tr>
<td>71</td>
<td>LOAD X72 WITH 8</td>
</tr>
<tr>
<td>72</td>
<td>LOAD X73 WITH 8</td>
</tr>
<tr>
<td>73</td>
<td>LOAD X74 WITH 8</td>
</tr>
<tr>
<td>74</td>
<td>LOAD X75 WITH 8</td>
</tr>
<tr>
<td>75</td>
<td>LOAD X76 WITH 8</td>
</tr>
<tr>
<td>76</td>
<td>LOAD X77 WITH 8</td>
</tr>
<tr>
<td>77</td>
<td>LOAD X78 WITH 8</td>
</tr>
<tr>
<td>78</td>
<td>LOAD X79 WITH 8</td>
</tr>
<tr>
<td>79</td>
<td>LOAD X80 WITH 8</td>
</tr>
<tr>
<td>80</td>
<td>LOAD X81 WITH 8</td>
</tr>
<tr>
<td>81</td>
<td>LOAD X82 WITH 8</td>
</tr>
<tr>
<td>82</td>
<td>LOAD X83 WITH 8</td>
</tr>
<tr>
<td>83</td>
<td>LOAD X84 WITH 8</td>
</tr>
<tr>
<td>84</td>
<td>LOAD X85 WITH 8</td>
</tr>
<tr>
<td>85</td>
<td>LOAD X86 WITH 8</td>
</tr>
<tr>
<td>86</td>
<td>LOAD X87 WITH 8</td>
</tr>
<tr>
<td>87</td>
<td>LOAD X88 WITH 8</td>
</tr>
<tr>
<td>88</td>
<td>LOAD X89 WITH 8</td>
</tr>
<tr>
<td>89</td>
<td>LOAD X90 WITH 8</td>
</tr>
<tr>
<td>90</td>
<td>LOAD X91 WITH 8</td>
</tr>
<tr>
<td>91</td>
<td>LOAD X92 WITH 8</td>
</tr>
<tr>
<td>92</td>
<td>LOAD X93 WITH 8</td>
</tr>
<tr>
<td>93</td>
<td>LOAD X94 WITH 8</td>
</tr>
<tr>
<td>94</td>
<td>LOAD X95 WITH 8</td>
</tr>
<tr>
<td>95</td>
<td>LOAD X96 WITH 8</td>
</tr>
<tr>
<td>96</td>
<td>LOAD X97 WITH 8</td>
</tr>
<tr>
<td>97</td>
<td>LOAD X98 WITH 8</td>
</tr>
<tr>
<td>98</td>
<td>LOAD X99 WITH 8</td>
</tr>
<tr>
<td>99</td>
<td>LOAD X100 WITH 8</td>
</tr>
<tr>
<td>100</td>
<td>LOAD X101 WITH 8</td>
</tr>
</tbody>
</table>
SUBPROGRAM NO. 3 - DATA MANIPULATION SUBROUTINES

CALCULATE SUBRoutines

33 CALC SAVE

00102 00104710000 010
00103 00266630000 010
00104 00266750000 010
00105 00266741000 010
00106 000030 4330 00 010
00107 000036 4630 00 010
00110 000036 5670 00 010
00111 000032 4770 00 010
00112 00262 4630 00 010
00113 21600 4350 03 000
00114 000141 7560 00 010
00115 00121 6040 00 010
00116 00020 2350 07 000
00117 000144 7550 00 010
00120 000129 7100 00 010
00121 000052 2350 07 000
00122 000117 7100 00 010
00123 000052 2350 07 000
00124 000136 7100 00 010
00125 000030 4330 00 010
00126 000032 5770 00 010
00127 000034 5670 00 010
00130 000036 5770 00 010
00131 00262 4630 00 010
00132 21600 4350 03 000
00133 000142 7560 00 010
00134 000123 6040 00 010
00135 000020 2350 07 000
00136 000145 7550 00 010
00137 000143 0940 00 010
00138 00013710000 010
00139 0000141 0000000000 000
00140 0000142 0000000000 000
00141 0000143 0000000000 000
00142 0000144 0000000000 000
00143 0000145 0000000000 000
00144 0000146 0000000000 000
00145 0000147 0000000000 000

DFLD AFLCAT
DFPP AFLCAT
DFFP =103
UFA =1825,DU
STG EBIT
THI EMINUS
LDA =C20,IL
STA E10N
TRA CONG
LDA =C52,IL
TRA STORE
FMINUS LDA =C52,IL
TRA STORE1
D10E DFLD AFLCAT
DFSB AFLCAT
DFFP =103
UFA =1825,DU
STG FS1N
FMINUS
LDA =C20,IL
STORE1 STA FS1N
AOS SEG1N
RETURN CALC
EB1N DEC 0
FB1N DEC 0
EB1N DEC 0
FS1N DEC 0
SEG1N DEC 0
FS1N DEC 0
AOS SEG1N
RETURN CALC
SUBPROGRAM NO. 3 - DATA MANIPULATION SUBROUTINES

LET'S CONVERT TO EIGHTARY

67 TOPYN SAVE

00146 0001500110000 010
00147 0002500630000 010
00150 0025000199000 010
00151 0002600471000 010
00152 0002602000000 010
00153 0002607000000 010
00154 0002607000000 010
00155 0002607000000 010
00156 0002607000000 010
00157 0002607000000 010
00158 0002607000000 010
00159 0002607000000 010
00160 0002607000000 010
00161 0002607000000 010
00162 0002607000000 010
00163 0002607000000 010
00164 0002607000000 010
00165 0002607000000 010
00166 0002607000000 010
00167 0002607000000 010
00168 0002607000000 010
00169 0002607000000 010
00170 0002607000000 010
00171 0002607000000 010
00172 0002607000000 010
00173 0002607000000 010
00174 0002607000000 010
00175 0002607000000 010
00176 0002607000000 010
00177 0002607000000 010
00178 0002607000000 010
00179 0002607000000 010
00180 0002607000000 010
00181 0002607000000 010
00182 0002607000000 010
00183 0002607000000 010
00184 0002607000000 010
00185 0002607000000 010
00186 0002607000000 010
00187 0002607000000 010
00188 0002607000000 010
00189 0002607000000 010
00190 0002607000000 010
00191 0002607000000 010
00192 0002607000000 010
00193 0002607000000 010
00194 0002607000000 010
00195 0002607000000 010
00196 0002607000000 010
00197 0002607000000 010
00198 0002607000000 010
00199 0002607000000 010
00200 0002607000000 010
00201 0002607000000 010
00202 0002607000000 010
00203 0002607000000 010
00204 0002607000000 010
00205 0002607000000 010
00206 0002607000000 010
00207 0002607000000 010
00208 0002607000000 010
00209 0002607000000 010
00210 0002607000000 010

68 LDA SUTAL3
69 STA TAL3
70 RTNX LDAQ RECWRK
72 LLS 12
74 SSTA RECWRK
76 TSX7 CONVTR
77 XEC STOREX,6
78 TRA *-3
79 AROWN LDA RECWRK*5
80 AAN A MASKA
81 LDX7 CLT,DL
82 EAX7 OUT
83 TRA CONVTR+1
84 CONVTR AAN A MASKA
85 TRA 0,7
86 STOREX LDA RECWRK*1
87 LDA RECWRK*2
88 LDA RECWRK*4
89 TRA AFOLN
90 OUT RETURN TOBIN
91 MASKA ECC1 1717171717
92 OAI 0000000000
93 MWRK WSL 1
94 TAL1 TALLY MANTX,6,0
95 TAL2 TALLY MANTX,6,0
96 TAL3 TALLY AEIA,4
97 SUTAL3 TALLY AEIA,4
NOW HOW ABOUT BCD?

000211  99 TOBCD SAVE

000212
000213
000214
000215
000216
000217
000218
000219
000220
000221
000222

000211  000213710000  010  LDX7 ,CU
000212  000260630000  010  LDA SEQ$IN
000213  000260754000  010  RFI 3,1
000214  000260741000  010  BCD CN79,7
000215  000000 2270 03  000  100  STG SEQ$IN
000216  000143 2350 00  010  101
000217  006200 5202 01  000  102
000218  000000 5050 17  000  103
000219  000257 7960 00  010  104
000220  000212710000  010  105  RETURN TOBCD
EDIT SUBROUTINE

107 DOIT MACRO
108 LDAQ #1
109 LLS #2
110 STA #1
111 LLS 6
112 C4S 6
113 CRL =C3, 000000
114 LRS 6
115 CRG #2
116 STC #2
117 ENDM DOIT
118 MACR MACRC
119 LDA =TAR #6
120 STA #5
121 LNG =CO2C202C202C
122 #4 LDA #1,SC
123 #5 TML #6
124 STC #2,SC
125 STC #3,SC
126 TRA #4
127 #6 LDG =PTRA #7
128 STC #5
129 #7 STA #6,SC
130 STA #7,SC
131 TTR #4
132 ENDM MACR
133 EDIT SAVE

000223 000225710000 010 000223
000224 000226053000 010 000225 0002260754000 010 000226 0002260741000 010
000227
000228 0002253270 00 010 000229 000226143000 010 000230 0002260754000 010
000231 0002260741000 010
000227
000232 0002253270 00 010 000233 000226143000 010 000234 0002260754000 010
000235 0002260741000 010
000227
000236 0002253270 00 010 000237 000226143000 010 000238 0002260754000 010
000239 0002260741000 010
000227
000240
000241 000226143000 010 000242 0002260754000 010 000243 0002260741000 010
000227
000244 000226143000 010 000245 0002260754000 010 000246 0002260741000 010
000227
000247 000226143000 010 000248 0002260754000 010 000249 0002260741000 010
000227
000250 000226143000 010 000251 0002260754000 010 000252 0002260741000 010
000227
000253 000226143000 010 000254 0002260754000 010 000255 0002260741000 010
000227
000256 000226143000 010 000257 0002260754000 010 000258 0002260741000 010
000227
000259 000226143000 010 000260 0002260754000 010 000261 0002260741000 010
000227
000262 000226143000 010 000263 0002260754000 010 000264 0002260741000 010
000227
000265 000226143000 010 000266 0002260754000 010 000267 0002260741000 010
000227
000268 000226143000 010 000269 0002260754000 010 000270 0002260741000 010
000227
EDIT SUBROUTINE

000002 202020202020 000
000003 202020202020 000
000004 202020202020 000
000005 202020202020 000
000006 202020202020 000
000007 202020202020 000
000008 202020202020 000
000009 202020202020 000
000010 202020202020 000
000011 202020202020 000
000012 202020202020 000
000013 202020202020 000
000014 202020202020 000
000015 202020202020 000
000016 202020202020 000
000017 202020202020 000
000018 202020202020 000
000019 202020202020 000
000020 202020202020 000
000021 202020202020 000
000022 202020202020 000
000023 202020202020 000
000024 202020202020 000
000025 202020202020 000
000026 202020202020 000
000027 202020202020 000
000028 202020202020 000
000029 202020202020 000
000030 202020202020 000
000031 202020202020 000
000032 202020202020 000
000033 202020202020 000
000034 202020202020 000
000035 202020202020 000
000036 202020202020 000
000037 202020202020 000
000038 202020202020 000
000039 202020202020 000
000040 202020202020 000
000041 202020202020 000
000042 202020202020 000
000043 202020202020 000
000044 202020202020 000
000045 202020202020 000
000046 202020202020 000
000047 202020202020 000
000048 202020202020 000
000049 202020202020 000
000050 202020202020 000
000051 202020202020 000
000052 202020202020 000
000053 202020202020 000
000054 202020202020 000
000055 202020202020 000
000056 202020202020 000
000057 202020202020 000
000058 202020202020 000
000059 202020202020 000
000060 202020202020 000
000061 202020202020 000

ERROR LINKAGE

000260 003000000000 000
000261 244346216320 000

LITERALS

000262 024764000000 000
<table>
<thead>
<tr>
<th>OCTAL</th>
<th>SYMBOL</th>
<th>REFERENCES BY ALTER NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>ALN</td>
<td>20 10 20 96 97</td>
</tr>
<tr>
<td>30</td>
<td>AFLOAT</td>
<td>24 14 24 34 49</td>
</tr>
<tr>
<td>10</td>
<td>AGAIN</td>
<td>9 9 18</td>
</tr>
<tr>
<td>135</td>
<td>AROUND</td>
<td>79 79 89</td>
</tr>
<tr>
<td>34</td>
<td>FFLOAT</td>
<td>25 25 37 50</td>
</tr>
<tr>
<td>112</td>
<td>CALC</td>
<td>33 33 60</td>
</tr>
<tr>
<td>34</td>
<td>CFLOAT</td>
<td>26 26 36 51</td>
</tr>
<tr>
<td>1/2</td>
<td>CCWRT</td>
<td>84 75 83 84</td>
</tr>
<tr>
<td>36</td>
<td>CFLOAT</td>
<td>27 27 35 52</td>
</tr>
<tr>
<td>1/3</td>
<td>CMODE</td>
<td>49 44 49</td>
</tr>
<tr>
<td>272</td>
<td>ERRCD</td>
<td>138 134 138</td>
</tr>
<tr>
<td>141</td>
<td>EER</td>
<td>81 40 61</td>
</tr>
<tr>
<td>2/3</td>
<td>ELET</td>
<td>133 133 137</td>
</tr>
<tr>
<td>1/2</td>
<td>EPINS</td>
<td>49 41 45</td>
</tr>
<tr>
<td>144</td>
<td>ESIGN</td>
<td>64 43 64 134</td>
</tr>
<tr>
<td>254</td>
<td>FFCOD</td>
<td>139 136 139</td>
</tr>
<tr>
<td>182</td>
<td>FREN</td>
<td>62 55 62</td>
</tr>
<tr>
<td>1/3</td>
<td>FFLOAT</td>
<td>4 4 49</td>
</tr>
<tr>
<td>145</td>
<td>FSIGN</td>
<td>65 58 65 136 99 133</td>
</tr>
<tr>
<td>260</td>
<td>F.E.L.</td>
<td>6 4 33 67 99 133</td>
</tr>
<tr>
<td>212</td>
<td>FASKA</td>
<td>91 80 84 91</td>
</tr>
<tr>
<td>204</td>
<td>FVKRAK</td>
<td>93 93 94 95</td>
</tr>
<tr>
<td>220</td>
<td>GLT</td>
<td>90 81 82 90</td>
</tr>
<tr>
<td>0</td>
<td>GWRK</td>
<td>143 71 74 86 87 88 143</td>
</tr>
<tr>
<td>180</td>
<td>REGLY</td>
<td>30 6 17 30</td>
</tr>
<tr>
<td>40</td>
<td>SAVREG</td>
<td>29 29 30 31</td>
</tr>
<tr>
<td>257</td>
<td>SEQRCD</td>
<td>141 104 141</td>
</tr>
<tr>
<td>143</td>
<td>SEGGIN</td>
<td>63 59 63 101</td>
</tr>
<tr>
<td>116</td>
<td>STORE1</td>
<td>58 48 58</td>
</tr>
<tr>
<td>1/3</td>
<td>STORE2</td>
<td>86 76 86</td>
</tr>
<tr>
<td>117</td>
<td>STORE</td>
<td>43 43 46</td>
</tr>
<tr>
<td>210</td>
<td>SVTAZ</td>
<td>97 68 97</td>
</tr>
<tr>
<td>2/7</td>
<td>TALT</td>
<td>96 69 96</td>
</tr>
<tr>
<td>1/1</td>
<td>TYPREG</td>
<td>31 5 31</td>
</tr>
<tr>
<td>211</td>
<td>TCRCL</td>
<td>99 99 105</td>
</tr>
<tr>
<td>146</td>
<td>THERIA</td>
<td>67 67 90</td>
</tr>
</tbody>
</table>

**16,957 WORDS OF MEMORY WERE USED BY GMAP FOR THIS ASSEMBLY.**
<table>
<thead>
<tr>
<th>ENTRY LOCATION</th>
<th>ENTRY LOCATION</th>
<th>ENTRY LOCATION</th>
<th>ENTRY LOCATION</th>
<th>ENTRY LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>011742</td>
<td>101167</td>
<td>011404</td>
<td>101271</td>
<td>011320</td>
</tr>
<tr>
<td>011320</td>
<td>101271</td>
<td>011320</td>
<td>101271</td>
<td>011320</td>
</tr>
</tbody>
</table>

**SUBPROGRAMS INCLUDED IN DECK:**

- `GETBK U07410`
- `GETET U07412`
- `GET U07412`
- `GETU U07412`
- `GETU6 U07412`
- `GETU10 U07412`
- `GETU12 U07412`
- `GETU2 U07412`
- `GETU4 U07412`
- `GETU6 U07412`

**SUBPROGRAMS OBTAINED FROM SYSTEM LIBRARY:**

- `GET BK U07410`
- `GET ET U07412`
- `GET U07412`
- `GET U6 U07412`
- `GET U10 U07412`
- `GET U12 U07412`
- `GET U2 U07412`
- `GET U4 U07412`
- `GET U6 U07412`
- `GET U10 U07412`

**ALLOCATED CORE OBJECT PROGRAM:**

- `SIZE 00000` `THRU 11177` `012000`
- `THRU 00350` `THRU 11177` `006430`
- `THRU 00100` `THRU 002002` `001703`
- `THRU 00350` `THRU 11177` `006430`
- `THRU 00100` `THRU 002002` `001703`
<table>
<thead>
<tr>
<th>FILE CODE</th>
<th>NAME</th>
<th>REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A*</td>
<td>ALTER FILE</td>
<td>Created when System Input (GIN) reads $UPDATE card. $ALTER cards and source cards put on A*.</td>
</tr>
<tr>
<td>B*</td>
<td>OBJECT PROGRAM FILE</td>
<td>Created by General Loader (GLOAD) when $EXECUTE included in source deck and no fatal error occurred during compilation. (General Loader loads from B* rather than from R* for compile-and-execute option)</td>
</tr>
<tr>
<td>C*</td>
<td>BINARY DECK FILE</td>
<td>Created when binary deck option ($GMAP DECK) is specified. Goes to SYSOUT--report code: 76.</td>
</tr>
<tr>
<td>G*</td>
<td>GMAP SOURCE FILE</td>
<td>System Input (GIN) creates G* when it sees $GMAP. COBOL &amp; FORTRAN create G* as output file. Used as input to GMAP Assembler.</td>
</tr>
<tr>
<td>H*</td>
<td>PROGRAM LINK FILE</td>
<td>Used for temporary storage of programs that are called into and out of memory.</td>
</tr>
<tr>
<td>I*</td>
<td>DATA STORAGE FILE</td>
<td>Used as storage area for input data required by User Programs.</td>
</tr>
<tr>
<td>K*</td>
<td>COMPRESSED DECK FILE</td>
<td>Created when COMDK option specified in language processor control card ($GMAP COMDK).</td>
</tr>
<tr>
<td>L*</td>
<td>SYSTEM SUBROUTINE LIBRARY FILE</td>
<td>Appropriate System Subroutines loaded as required.</td>
</tr>
<tr>
<td>P*</td>
<td>SYSTEM OUTPUT FILE</td>
<td>Collection of all Systems Output.</td>
</tr>
<tr>
<td>R*</td>
<td>LOADER INPUT FILE</td>
<td>Normal Loader Input File. Includes directives to General Loader (GLOAD) as well as binary programs. $SOURCE directive generated by system and placed on R* for compile-and-execute jobs instructs General Loader to load from B* rather than R*.</td>
</tr>
</tbody>
</table>
SERIES 6000

GMAP PROGRAMMING

COURSE 605

EXERCISES

<table>
<thead>
<tr>
<th>EXERCISE NO.</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Binary and Octal Representation</td>
<td>287</td>
</tr>
<tr>
<td>2</td>
<td>Data Movement and Arithmetic Operations</td>
<td>288</td>
</tr>
<tr>
<td>3</td>
<td>Data Representation and Shift Operations</td>
<td>289</td>
</tr>
<tr>
<td>4</td>
<td>Indirect then Tally Address Modification</td>
<td>290</td>
</tr>
<tr>
<td>5</td>
<td>Miscellaneous Instructions</td>
<td>291</td>
</tr>
<tr>
<td>6</td>
<td>Pseudo Operations</td>
<td>292</td>
</tr>
<tr>
<td>7</td>
<td>Desk Debug Exercises</td>
<td>293</td>
</tr>
<tr>
<td>8</td>
<td>Macro Operations</td>
<td>294</td>
</tr>
<tr>
<td>9</td>
<td>File and Record Control (GFRCC) Exercise No. 1</td>
<td>295</td>
</tr>
<tr>
<td>10</td>
<td>File and Record Control (GFRCC) Exercise No. 2</td>
<td>297</td>
</tr>
</tbody>
</table>

PROBLEMS

<table>
<thead>
<tr>
<th>PROBLEM NO.</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Workshop Problem</td>
<td>299</td>
</tr>
<tr>
<td>2</td>
<td>Submit Class Problem</td>
<td>310</td>
</tr>
<tr>
<td>3</td>
<td>Optional Problem No. 2</td>
<td>310</td>
</tr>
</tbody>
</table>

*Workshop Performance Chart* 311
1. Do the following number conversions:

   a. \((285)_{10} = (\quad)_8 = (\quad)_2\)
   b. \((.75)_{10} = (\quad)_8 = (\quad)_2\)
   c. \((65)_8 = (\quad)_{10} = (\quad)_2\)
   d. \((.101)_2 = (\quad)_8 = (\quad)_{10}\)
   e. \((1101)_2 = (\quad)_8 = (\quad)_{10}\)
   f. \((.31)_8 = (\quad)_{10} = (\quad)_2\)
   g. \((1011011)_2 = (\quad)_8 = (\quad)_{10}\)
   h. \((8)_{10} = (\quad)_8 = (\quad)_2\)

2. Do the following arithmetic operations:

   a. \((1011)_2 + (111)_2 = (\quad)_2\)
   b. \((23)_8 + (57)_8 = (\quad)_8\)
   c. \((1110)_2 - (11)_2 = (\quad)_2\)
   d. \((6307)_8 - (1436)_8 = (\quad)_2\)
   e. \((111011)_2 - (11001)_2 = (\quad)_2\)
   f. \((110)_2 + (11)_2 + (101)_2 + (111)_2 = (\quad)_2\)
DATA MOVEMENT AND ARITHMETIC OPERATIONS

1. Move 150 words from "Table A" to "Table B." Check each word and:
   a. If negative, add one to "NEG" counter.
   b. If zero, add one to "ZERO" counter.
   Update your instruction word addresses without using register modification.

2. Repeat the problem for Step #1, except reverse the order of the Table
   and use register address modification.

3. Show the coding to complement a negative number back into its true form.
   Do not use the "NEG" instruction.

4. Show the coding to perform the following calculation:

   \[
   \frac{(A \cdot B) \ - \ C}{D} \ + \ E = F
   \]

   Note: If D equal to zero, store zero's in "F."
DATA REPRESENTATION AND SHIFT OPERATIONS

1. Create the following values in memory by means of pseudo operations:
   A. $1492_{10}$
   B. $3466_8$
   C. $1903_{10}$ in upper 18 bits
   D. $770077_8$ in upper 18 bits
   E. $3.1416_{10}$
   F. $7.9 \times 10^3$
   G. $83.5 \times 10^9$ in double precision
   H. $183 \times 10^{-3}$ scaled to Bit 24
   I. 6 bits of BCD X, 6 bits octal 72, 6 bits of $28_{10}$, 18 bits of address XRAY
   J. The statement "THREE-D SYSTEM"

2. Shift the binary number in memory location "Place" three binary places to the left and store in "New"
   A. Question: What effect does this operation have on this value?

3. Add "A" ($= 14.73B23$) to "B" ($= 6.097B27$) and store in Location "C"

4. Show the control cards and deck setup necessary to perform the following jobs:
   A. GMAP assembly and execute
   B. GMAP and FORTRAN assembly and execute
   C. Binary object deck and execute
INDIRECT THEN TALLY ADDRESS MODIFICATION

1. Code Problems 1 and 2 of Exercise #2 using Tally operations.

2. There are ten (10) words of 6-bit BCD characters beginning at Location "TABLEA." Completely reverse the characters so that the first character of the first word becomes the last characters of the last word and so on. Store result in TABLEB.

Write the GMAP coding to do the following:

3. Cause a fault trap to occur.

4. Load A with B indirect.

5. Move 5 characters starting with character position 3 in TAB to PRTOUT starting in character position 5. Use tally modification.

6. Move a 50 word table (TAB1) to TAB2 using tallies.

7. Do part 6 using LDAQ, STAQ.

8. Give the major difference between CI and SC modification.
MISCELLANEOUS INSTRUCTIONS

1. Code Problem #1 of Exercise #2 using a Repeat type instruction.

2. Convert a binary value in Location "BIN" to BCD and store in "BCDNR."
   The maximum size of the binary number will not exceed 7 BCD digits when converted.

3. A physical record of 1000 words starts at Location "WORD" and is composed of 10-word logical records. If the first word of a logical record is negative, the logical record is to be moved to a second table starting at Location "WORK."

4. Write the GMAP coding to do the following:
   a. Move a 50 word table using indexing from TAB1 to TAB2.
   b. Do part "a" using RPD.
   c. Find the data record with a key of 2AB in a 50 record chain starting at MASCHN.
   d. Count all numbers <=0 or >=50 in table TAB which is 100 words long.

5. Show the contents of the instruction word.
   a. RPT 10,1
   b. RPDA 25,2,TZE,TMI
List the pseudo-operation which will do the following:

1. Reserve 500 words of core with the first word labeled A1.
2. Reserve 500 words of core with word 501 labeled A2.
3. Program documentation
4. Page title
5. Termination of assembly
6. Creates entry points
7. Used to begin a subroutine
8. Specifies labeled or blank common mode.
**Honeywell**

**SYMBOLOGIC CODING FORMS**

**GMAP STUDENT EXERCISE #7**

**Problem**

Find the Coding Errors only and describe them in the Comments Field.

**Programmer**

Do not concern yourself with possible programming logic errors.

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>E</th>
<th>OPERATION</th>
<th>ADDRESS, MODIFIER</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 6 7 8</td>
<td>14 16</td>
<td>2358</td>
<td>ORG 2358</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GOOD</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>.OCT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ONE, X7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2286</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td>29, 3, TZE, TRA</td>
<td>RPD 29, 3, TZE, TRA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>REG, 6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OCT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>IC-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>HI</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>HALT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HELP</td>
<td></td>
<td>2, 0 ,DU</td>
<td>LX 2, 0, DU</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>24, 7</td>
<td>RAL 24, 7</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>BYE, DU</td>
<td>ASQ BYE, DU</td>
<td></td>
</tr>
<tr>
<td>GOOD</td>
<td>0</td>
<td>REG</td>
<td>SREG REG</td>
<td></td>
</tr>
<tr>
<td>A=3</td>
<td></td>
<td>HELP</td>
<td>TMT HELP</td>
<td></td>
</tr>
<tr>
<td>HALT</td>
<td></td>
<td>GESTOP</td>
<td>MME GESTOP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>712345</td>
<td>ORG 712345</td>
<td></td>
</tr>
<tr>
<td>GOOD</td>
<td>0</td>
<td>BCI</td>
<td>BCI 4, I hope you got this correct.</td>
<td></td>
</tr>
<tr>
<td>REG</td>
<td>7</td>
<td>8</td>
<td>BS SS 8</td>
<td></td>
</tr>
<tr>
<td>BYE</td>
<td></td>
<td>HELP</td>
<td>ZERO HELP,</td>
<td></td>
</tr>
<tr>
<td>HI</td>
<td></td>
<td>66724</td>
<td>DEC 66724</td>
<td></td>
</tr>
<tr>
<td>OCT</td>
<td></td>
<td>123456787654</td>
<td>OCT 123456787654</td>
<td></td>
</tr>
<tr>
<td>END</td>
<td></td>
<td>END</td>
<td>END</td>
<td></td>
</tr>
</tbody>
</table>
MACRO OPERATIONS

1. Write a conditional assembly MACRO to move a variable number of words from one location to another. (Range 1-4095 words to be moved.) Also provide the option to either check and count ZERO and NEGATIVE words, or to ignore them. (Use the conditional PSEUDO OPS.)

2. Write a MACRO to move n words from argument #2 to argument #3. N is argument #1. Write an example of the CALL.

3. Write a MACRO to: swap N words from two given locations. Write an example of the CALL.
1. Write the coding for the File Control Block, File Designator Word, and Buffers for an input tape file with the following parameters:

   Fixed length records
   Record size 60 words
   Block size 300 words
   Two buffers
   File Code AB
   Standard labels
   Multireel file
   File name TAPE-FILE
   Location of File Control Block TPFILE
   Location of File Designator Word FILEWD

2. Write the coding for the File Control Block, File Designator Word, Buffers and Working Storage Area for an input BCD card file with the following parameters:

   File Code I*
   System Standard Format
   One Buffer
   Location of File Control Block CDFILE
   Location of File Designator Word CARDWD

3. Give the effective address of the following examples:

   a. LDA A,*
      A LDQ B,*5
      B LDX5 C,1

   b. LDA *+1,I
      LDQ *-1,*
3.

c.  LDX1  B,1*
    \  \  
   B+C(X_1) STX1  C,*1
    \  \  
     C  AIDX1  D,*5
     \  \  
      D  SBX1  10,DU
      \  \  
     1(+C(X_2)) AIDX1  15,DU

4. Form the one's complement of ABLE in BAKER.

5. Turn on bits 1, 13-15, and 27 and turn off all others in SAM.
1. Write the coding for the File Control Block, File Designator Word, Buffer, and Working Storage Area for an output card file with the following parameters:

   File Code P*
   Standard System Format
   One Buffer
   Location of File Control Block PUNCD
   Location of File Designator Word PDWD

2. Using the card file described in No. 1 above, write the following File and Record Control (GFRC) CALL's:

   a. OPEN
   b. CLOSE

3. Write the coding for the File Control Block, File Designator Word, Buffer for an output tape file with the following parameters:

   File Code TO
   Variable Length Records
   Record Size 50 words
   Block Size 200 words
   Two Buffers
   Location of File Control Block OTFILE
   Location of File Designator Word OTWD

4. Using the tape file described in No. 3 above, write the following File and Record Control (GFRC) CALL's.

   a. OPEN
   b. CLOSE
   c. PUT

5. Using CALL WTREC, write the coding to punch a binary card and other required parameters.

6. Using CALL WTREC, write the coding to punch a Hollerith card and any other required parameters.

7. Using CALL WTREC, write the coding and any other required parameters to print a 20-word line.
8. Using CALL WTREC, write the coding and any other required parameters to create a tape record of 15 words.

9. Using CALL PUNCH, write the coding and any other required parameters to punch a binary card.

10. Using CALL PUNCH, write the coding and any other parameters to punch a Hollerith card.

11. Using CALL PRINT, write the coding and any other parameters to write a 22-word line. Skip to the top of page after printing.

12. Write the IOEDIT control parameters and codes to insert 30 characters in the first line of the heading.

13. Write the IOEDIT control parameters and codes to print a sub-heading.

14. Write the IOEDIT control parameters and codes to insert a sequence number in punched cards beginning with 00000100.
GMAP WORKSHOP PROBLEM
(Optional)

PROBLEM DEFINITION

The GMAP Workshop problem consists of reading an input file of data cards, performing calculations on fields contained in these cards and printing the results on a report.

The input cards have four data fields labeled A, B, C & D respectively and a text field. The numeric input data from fields A, B, C & D must be converted from BCD values to binary values for calculations. After performing the calculations on these four fields, the results should be converted to BCD for output. The text field is then moved intact to the output areas.

Print line images should be edited by inserting decimal points and suppressing leading zeroes. These results should be formatted as specified.

The program solution consists of three subprograms which form one program. Sub-program ONE is the main or executive routine that consists solely of calls to the subroutine modules contained in subprogram TWO and subprogram THREE. The sequence of the calls inserted into subprogram ONE determines the sequential order of processing at execution time. This process is repeated until all data cards have been processed and the job is terminated.
GMAP COURSE

WORKSHOP PROBLEM JOB STACK
(Assignment No. 1)

$ SNUMB
$ IDENT
$ GMAP
  .
  .
  .
  END
$ GMAP
  NDECK
  (program one)

SOURCE CARDS

$ INIT
$ INPUT
$ OUTPUT
$ WRAPUP
$ END
$ GMAP
  NDECK
  (program two)

SOURCE CARDS

$ TOBIN
$ CALC
$ TOBCD
$ EDIT
$ END
$ EXECUTE
$ LIMITS
$ ENDJOB
$ DUMP
,5K,,1000

***EOF
ASSIGNMENT NO. 2

RDWRK contains the input card image.
ASSIGNMENT NO. 3

1. Assume EBIN, FBIN, and SEQBIN single precision fixed point binary numbers with a binary point to right of bit 35.

2. ESIGN and FSIGN will contain the sign of E and F, respectively.
ASSIGNMENT NO. 4

1. EBCD and FBCD are double precision words.

2. SEQBCD is a single precision word.
EDIT

EDIT EBCD TO:
(EEEEEEE.EEEE)

EDIT FBCD TO:
(FFFFFFFF.FFFS)

ZERO SUPPRESS EBCD &
FBCD TO DECIMAL.
S = BLANK IF PLUS AND
(-) IF NEGATIVE

MOVE SEQBCD, EBCD &
FBCD & VARIABLE FIELD
TO PRTOUT. SEE DATA
FORMAT.

EXIT

ASSIGNMENT NO. 5
PRTOUT located in labeled common
IØ.WRK.
INITIALIZE FILES. SET UP PRINT HEADING AS PER FORMAT.

EXIT

INPUT

READ CARD TO RDWRK IF E.O.F. GO TO WRAPUP

EXIT

OUTPUT

PRINT BODY LINE FROM PRTOUT

EXIT

WRAPUP

CLOSE FILES MME GEFINI

ASSIGNMENT NO. 6

Perform all I/O functions using File and Record Control (GEFRC).
**Input Item Description**

**Input Card Format**

<table>
<thead>
<tr>
<th>12</th>
<th>3--8</th>
<th>9--12</th>
<th>13--18</th>
<th>19--24</th>
<th>25--30</th>
<th>31--80</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;A&quot;</td>
<td>&quot;B&quot;</td>
<td>&quot;C&quot;</td>
<td>&quot;D&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**VARIABLE-LENGTH FIELD "V"**

**50 CHARACTERS MAXIMUM**

**Numeric Data Fields**

**Alphanumeric Data Fields**

**Description**

1. Input item is in BCD.
2. Numeric data fields contain positive integers; field "C" does not contain zero.
3. Each numeric data field is right-justified within its own area. It does not contain leading zeroes.
4. Variable-length field "V" contains some type of alphanumeric descriptive material. If present, field "V" begins with Column 31.
Printed Report Format and Description

The printed report will consist of the following:

1. Heading line - Any desired information
2. Sub-heading line - Any desired information
3. Body lines - The body line should be double-spaced. The print positions for each of the fields in the body line is as follows:

<table>
<thead>
<tr>
<th>Char. position</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 9</td>
<td>Blank</td>
</tr>
<tr>
<td>10 – 12</td>
<td>Seq. No.</td>
</tr>
<tr>
<td>13 – 18</td>
<td>Blank</td>
</tr>
<tr>
<td>19 – 30</td>
<td>EBCD</td>
</tr>
<tr>
<td>31 – 36</td>
<td>Blank</td>
</tr>
<tr>
<td>37 – 48</td>
<td>FBCD</td>
</tr>
<tr>
<td>49 – 54</td>
<td>Blank</td>
</tr>
<tr>
<td>55</td>
<td>Variable Field</td>
</tr>
</tbody>
</table>
CLASS PROBLEM - JOHN SMITH

AUG. 14, 1972

2:30 P.M.

FIRST OUTPUT
EXAMPLE
THIS MAY BE 50 CHAR.
NO DECIMAL POINT USED

Field "V"
Field "F"
OPTIONAL PROBLEM NO. 1

SUMBIT CLASS PROBLEM

1. BLANK THE PRINT AREA USING INDEXING.

2. USING FILE AND RECORD CONTROL (GFRC)
   a. OPEN THE FILES
   b. USE CALL GET AND MOVE THE RECORD TO WORKING STORAGE.
   c. EOF ENCOUNTERED CLOSE FILES.
   d. USE IMMEDIATELY.

3. CHECK THE FIRST WORD OF WORKING STORAGE FOR CODE SUMBIT.

4. CODE NOT EQUAL TO SUMBIT.
   a. MOVE THE RECORD TO THE PRINT AREA.
   (USE THE DOUBLE REPEAT FOR THE MOVE)
   b. MOVE AN ERROR MESSAGE TO THE PRINT AREA.
   c. USE CALL PRINT TO PRINT THE RECORD.

5. CODE EQUAL TO SUMBIT.
   a. COUNT THE BITS SET TO ONE IN WORDS 3 AND 4.
   b. MOVE THE RECORD TO THE PRINT AREA (USING TALLYS).
   c. CONVERT THE SUM OF BITS FROM WORDS 3 AND 4 TO BCD.
   d. MOVE BCD SUM TO PRINT AREA.
   e. PRINT THE VALID RECORD WITH THE SUM USING CALL PRINT.

OPTIONAL PROBLEM NO. 2

Given values A, B, and C (all values 1)

IF  \( A = B = C \) then \( X = A + B + C \)

IF  \( A = B \) or \( B = C \) or \( A = C \) then add the two equal values and multiply by the third = Y

IF  \( A \neq B \neq C \) then divide largest by smallest and subtract middle = Z.
GMAP COURSE

WORKSHOP PERFORMANCE CHART

NAME: ____________________________
CLASS NO.: ________________________

WORKSHOP PROBLEM

<table>
<thead>
<tr>
<th>TOBIN</th>
<th>CODED</th>
<th>DEBUGGED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CALC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOBCD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INPUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUTPUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRAPUP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIONAL PROBLEM NO. 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIONAL PROBLEM NO. 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

EXERCISES:

<table>
<thead>
<tr>
<th>#1</th>
<th></th>
<th>#6</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>#2</td>
<td></td>
<td>#7</td>
<td></td>
</tr>
<tr>
<td>#3</td>
<td></td>
<td>#8</td>
<td></td>
</tr>
<tr>
<td>#4</td>
<td></td>
<td>#9</td>
<td></td>
</tr>
<tr>
<td>#5</td>
<td></td>
<td>#10</td>
<td></td>
</tr>
</tbody>
</table>

MACRO

<table>
<thead>
<tr>
<th>MACRO</th>
<th></th>
<th>BOOL</th>
<th></th>
<th>SYMDEF</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>INE/IFE/IFL/IFG</td>
<td></td>
<td>OCT</td>
<td></td>
<td>SYMREF</td>
<td></td>
</tr>
<tr>
<td>IDR</td>
<td></td>
<td>VFD</td>
<td></td>
<td>EQU</td>
<td></td>
</tr>
<tr>
<td>RPT</td>
<td></td>
<td>DEC</td>
<td></td>
<td>BLOCK</td>
<td></td>
</tr>
<tr>
<td>PMC</td>
<td></td>
<td>BCI</td>
<td></td>
<td>USE</td>
<td></td>
</tr>
<tr>
<td>TTL/TTLS</td>
<td></td>
<td>ARG</td>
<td></td>
<td>SET</td>
<td></td>
</tr>
</tbody>
</table>

311
<table>
<thead>
<tr>
<th>QUIZ NO.</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Overview and Data Quiz</td>
<td>313</td>
</tr>
<tr>
<td>2</td>
<td>Instruction Counter</td>
<td>315</td>
</tr>
<tr>
<td>3</td>
<td>Tally Operations</td>
<td>316</td>
</tr>
<tr>
<td>4</td>
<td>Miscellaneous Operations</td>
<td>317</td>
</tr>
<tr>
<td>5</td>
<td>General Quiz</td>
<td>318</td>
</tr>
<tr>
<td>6</td>
<td>Macro Operations</td>
<td>322</td>
</tr>
<tr>
<td>7</td>
<td>Effective Address and File and Record Control (GFRC) Quiz</td>
<td>323</td>
</tr>
<tr>
<td>8</td>
<td>File and Record Control (GFRC) Examination</td>
<td>324</td>
</tr>
<tr>
<td>9</td>
<td>GMAP Quiz 9</td>
<td>326</td>
</tr>
<tr>
<td>10</td>
<td>GMAP Quiz 10</td>
<td>335</td>
</tr>
<tr>
<td></td>
<td>Final Examination</td>
<td>341</td>
</tr>
</tbody>
</table>
1. Name the three component modules of the Series 6000.

________________, ______________, ______________.

2. Name six peripheral devices available with the Series 6000.

________, __________, __________, __________,

________, __________.

3. Name the two processor modes: ______________ and ______________.

4. Name five processor registers excluding index registers.

____________, ______________, ______________,

____________, ______________.

5. Maximum memory size on the Series 6000 is ___________ K.
6. Fill in missing information as it will appear on the listing when the following pseudo operations are assembled by GMAP.

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>INSTRUCTION</th>
<th>CARD</th>
<th>IMAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>000600</td>
<td></td>
<td>DATA</td>
<td>ØCT</td>
</tr>
</tbody>
</table>

| BCI       | 2, SEPTEMBER |
| VFD       | 18/DATA,H12/CR |
| VFD       | H36/ASEA |
| VFD       | Ø24/74,6/2,6/-1, |
| ETC       | 36/40,2/,4/-6 |
| DEC       | -4,1,+3,2B27 |
| DEC       | 1B71,-8B63,100B68 |
INSTRUCTION COUNTER

Fill in missing information as it would appear after the execution of each instruction in the order of execution.

<table>
<thead>
<tr>
<th>OCTAL LOCATION</th>
<th>INSTRUCTION</th>
<th>MEMORY OR REGISTER OCTAL CONTENTS AFTER EXECUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>000500</td>
<td>START</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LDX5 5,DU</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LDX7 -7,DU</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LDX2 -7,DL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LDA 2,DU</td>
<td></td>
</tr>
<tr>
<td>000601</td>
<td>BETA</td>
<td></td>
</tr>
<tr>
<td>000602</td>
<td>TRA ALPHA</td>
<td></td>
</tr>
<tr>
<td>000603</td>
<td>STC1 SAVE1</td>
<td></td>
</tr>
<tr>
<td>000604</td>
<td>TSX3 TEST</td>
<td></td>
</tr>
<tr>
<td>000605</td>
<td>STC2 SAVE2</td>
<td></td>
</tr>
<tr>
<td>000606</td>
<td>TRA J</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA *+1</td>
<td></td>
</tr>
<tr>
<td>000700</td>
<td>ALPHA</td>
<td></td>
</tr>
<tr>
<td>000701</td>
<td>LDI RGR</td>
<td></td>
</tr>
<tr>
<td>000702</td>
<td>STI SAVE1</td>
<td></td>
</tr>
<tr>
<td>000703</td>
<td>LDX2 RGR</td>
<td></td>
</tr>
<tr>
<td>000704</td>
<td>TRA BETA+1</td>
<td></td>
</tr>
<tr>
<td>000705</td>
<td>RGR OCT 000200404000</td>
<td></td>
</tr>
<tr>
<td>000706</td>
<td>SAVE1 ZERO</td>
<td></td>
</tr>
<tr>
<td>000707</td>
<td>TEST</td>
<td></td>
</tr>
<tr>
<td>000710</td>
<td>TRA 0,3</td>
<td></td>
</tr>
<tr>
<td>000711</td>
<td>SAVE2 ZERO</td>
<td></td>
</tr>
<tr>
<td>000712</td>
<td>J RET SAVE2</td>
<td></td>
</tr>
</tbody>
</table>
TALLY OPERATIONS

1. Sixty characters are stored in ten words in memory beginning at location DATA. Store the characters, one character to a word, in right justified form with leading zeros in a sixty-word table beginning at location SPRED.

2. You are given a table of 100 one-word values stored in memory beginning at MAYBZ. Move all nonzero values to a table starting at NOZER. Put the number of nonzero values you have found into memory cell NOZER-1.

3. Write the octal representation of the value resulting from each of the following pseudo operations:

   a. TALLY   1024,48
   b. TALLYD  384,40,4
   c. TALLY   2048,18,3
MISCELLANEOUS OPERATIONS

1. TRANS contains a one digit BCD code in right justified form.
   Write the coding which will test the code for a range of 0 thru 9.
   If not 0 thru 9, branch to ERROR. If it is one of the digits 0 thru 9, branch to UPDATE.

2. Location BRANCH contains a TRA 0. If location CODE contains a 1, set the instruction at BRANCH to transfer to RETURN. If CODE contains a 3, set the instruction to transfer to CALC.

3. A one word field called CODE normally contains only positive BCD numeric characters. Check the field for the presence of non-numeric characters. If any are found, branch to ERROR.

4. Set the 800 words of memory beginning at location BUILD to zero.

5. Sum the 100 words stored in TABLE and store the sum in SIGMA.
GENERAL QUIZ

SECTION I

This section of the quiz consists of True-False items, each of which has a value of one (1) point. Enter your answer for each item on this sheet, making sure that you check the correct line for the corresponding test item.

1. The BLOCK pseudo-op may be used to assemble data and/or instructions into blank common.

2. The TZE instruction will transfer and turn off the zero indicator if it is on.

3. The expression (Y-pair) means any two sequential memory locations that begin at an even address.

4. The contents of an index register may be incremented or decremented.

5. LDX 3,4 will load X3 with contents of Location 4.

6. LDx4 3,DL will load X4 with zeros.

SECTION II

This section of the quiz consists of multiple choice items, each of which has a value of two (2) points. Each wrong answer will deduct two points from the total. Enter your answer in the A, B, C, or D column on this sheet, making sure that you check the correct line for the corresponding test item.

NOTE! There is only one (1) correct answer for each item in this section. Answer sheets which contain more than one answer per item will be considered to be wrong!

1. At assembly time--

   LDA =MLDX1W5,DU

   A. Will generate an entry in the literal pool.
   B. The LDA is an instruction literal.
   C. The LDA is a direct operand instruction.
   D. Invalid coding.
2. **LDA -MTRAYLOC,5**

The above coding is an example of:

___ A. Direct upper (DU) address modification.
___ B. An instruction literal.
___ C. An instruction literal, but the ",5" is illegal.
___ D. An illegal operand.

3. Given the following coding example, what will be the contents of the A and Q registers after execution of the LDAQ instruction?

<table>
<thead>
<tr>
<th>Address</th>
<th>Opcode</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>000062</td>
<td>LDAQ</td>
<td>TABLE2</td>
</tr>
<tr>
<td>000063</td>
<td>TRA</td>
<td>OUT</td>
</tr>
<tr>
<td>000064</td>
<td>TABLE1</td>
<td>DEC 5</td>
</tr>
<tr>
<td>000065</td>
<td>DEC</td>
<td>6</td>
</tr>
<tr>
<td>000066</td>
<td>DEC</td>
<td>7</td>
</tr>
<tr>
<td>000067</td>
<td>TABLE2</td>
<td>DEC 1</td>
</tr>
<tr>
<td>000070</td>
<td>DEC</td>
<td>2</td>
</tr>
<tr>
<td>000071</td>
<td>DEC</td>
<td>3</td>
</tr>
</tbody>
</table>

___ A. A = 6, Q = 7
___ B. A = 7, Q = 1
___ C. A = 1, Q = 2
___ D. A = 2, Q = 3

4. Indicate the coding which will store characters 2, 4, and 5 from the Q register to the same character positions of location X.

___ A. STCQ X,13
___ B. STCQ X,26
___ C. STCQ X,31
___ D. None of the above.

5. **TABLE OCT -43**

The above line of coding will cause which of the following octal configurations to be generated?

___ A. 100000000043
___ B. 777777777735
___ C. 777777777743
___ D. 400000000043
6. LDA 5,DU

In this instruction, the number five is:

A. Found in bits 18 - 35 of the instruction itself.
B. Found in bits 0 - 17 of the instruction itself.
C. Memory location five.
D. Memory location five as modified by the DU directive.

7. LDA ,DU

The above instruction will cause which of the following octal contents to appear in bits 0 - 17 of the instruction?

A. 000023
B. 000000
C. 235003
D. 235007

8. STX2 0,DL

What, if anything, is wrong with this instruction?

A. Nothing is wrong with this coding.
B. The DL should be changed to a DU.
C. The zero operand is illegal in slave mode.
D. Direct type address modification is not allowed with a STX2.

SECTION III

This section of the quiz consists of completion type items, each of which has a value of three (3) points. Each wrong answer will deduct three points from the total.

NOTE! Each answer must be entirely correct to receive credit. No partial credit will be given (1 or 2 points) or items which are "almost" correct.

All answers are brief, and should be entered in the space provided on this sheet.

1. TSX1 30113

Prior to the execution of the above instruction, the IC contains value 11228 and the 30113 is an octal number. What will the IC contain (in octal) after execution?
2. Referring to the information and conditions set forth in Item (Question) #1, what will be the octal contents of index register one after execution?

3. Show the binary contents of a literal generated by: \[ = 1.5B7 \]
MACRO OPERATIONS

1. Write a MACRO procedure to:

   a. Clear N words

   b. Move N words from a given location to another given location.

   c. Swap N words from two given locations.

2. Write an example for the CALL of each of the above examples.
1. Give the effective addresses of the following instructions.

   a.       LDA       Cl,*
            LDQ       B5,*3
            LDX5      A,1
   
   b.       LDA       *+1,I
            LDQ       *-1,*
   
   c.       LDX1      B,1*
            B+C(X)    STX1      A,*1
            A         ADX1      F,*5
            F         SBX1      10,DU
            10+C(X)   ADX1      15,DU

2. Reverse the bits in the upper half of CHAR. That is

       if CHAR₁ = 1 turn it off
       if CHAR₁ = 0 turn it on

3. Show the coding to step through TAB until you find a zero word
   using index registers.

4. Define the following:
   a. File Control Block
   b. File Designator Word
   c. Buffer
FILE AND RECORD CONTROL (GFRC) EXAMINATION

The following statements are either true or false. Write in the blank on the left a T for true or F for false.

____ 1. A file must be opened before it can be read from or written on.

____ 2. The user may open all his files with the use of one OPEN command.

____ 3. If the user taries to OPEN a file that is already open, he will be aborted.

____ 4. If the OPEN routine determines that a file has been assigned to SYSOUT, but the file control block for the file does not indicate standard systems format, the user will be aborted.

____ 5. After executing a GET, the current record index (word 0 of the file control block) contains the location of the first data word of the requested logical record.

____ 6. When executing a GET, the EOF exit will be taken any time that any file mark is found.

____ 7. The user can obtain the previous logical record processed by using the GETBK command.

____ 8. The PUT command must be used any time the user desires to move a logical record from the input buffer to working storage.

____ 9. The PUTBK command can be used if the programmer desires to put his logical record at the beginning of the next buffer, regardless whether or not the current buffer is full.

____ 10. CALL PRINT always requires a CALL IOEDIT to have been previously executed.

____ 11. Every time a CALL GET or CALL PUT is executed, the physical device (tape, for example) will be moved.

____ 12. All the information in a file control block is stored there by the user via the FILCB macro.

____ 13. File and Record Control (GFRC) subroutines are loaded into memory by General Loader (GLOAD).
14. File and Record Control (GFRG) does not use GCOS.

15. The subset of File and Record Control (GFRG) used by the user resides in a special area within.

16. Standard magnetic tape labels are 14 words long.

17. Random disc/drum files must be processed using logical record processing commands.

18. In the standard system format a block serial number will exist as the first word of each block.

19. When output is to be stacked on SYSOUT, the user must allow for two extra words in the buffer(s).

20. The only type of records File and Record Control (GFRG) can handle are variable length records.

21. One file control block must be provided for each file used in the program.

22. Mixed density is not allowed within a magnetic tape file processed by File and Record Control (GFRG).

23. The OPEN and CLOSE routines obtain their orders from file designator words.

24. When buffers are used in conjunction with File and Record Control (GFRG) logical record processing commands, each buffer must be one word larger than the largest physical record it is to hold.

25. When using the command READ, the user must provide his own list of control words.
To the left of each function listed in column "A" write the number of the item listed in column "B" that performs this function.

<table>
<thead>
<tr>
<th>&quot;A&quot;</th>
<th>&quot;B&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary communication between the user program and File and Record</td>
<td>1. GETBK</td>
</tr>
<tr>
<td>Control (GFRC) subroutines</td>
<td>2. One Word greater than</td>
</tr>
<tr>
<td></td>
<td>physical record</td>
</tr>
<tr>
<td></td>
<td>3. Standard Labels</td>
</tr>
<tr>
<td></td>
<td>4. File Control Blocks</td>
</tr>
<tr>
<td></td>
<td>5. CALL</td>
</tr>
<tr>
<td></td>
<td>6. GET</td>
</tr>
<tr>
<td></td>
<td>7. PUT</td>
</tr>
<tr>
<td></td>
<td>8. PUTBK</td>
</tr>
<tr>
<td></td>
<td>9. COPY</td>
</tr>
<tr>
<td></td>
<td>10. READ</td>
</tr>
<tr>
<td></td>
<td>11. File designator word</td>
</tr>
<tr>
<td></td>
<td>12. File Control Cards</td>
</tr>
<tr>
<td></td>
<td>13. FLCB</td>
</tr>
<tr>
<td></td>
<td>14. OCT</td>
</tr>
<tr>
<td></td>
<td>15. OPEN</td>
</tr>
<tr>
<td></td>
<td>16. VFD</td>
</tr>
<tr>
<td></td>
<td>17. RDREC</td>
</tr>
<tr>
<td>Provides information to the OPEN and CLOSE routines</td>
<td></td>
</tr>
<tr>
<td>Whenever the logical record blocking/deblocking facilities of File</td>
<td></td>
</tr>
<tr>
<td>and Record Control (GFRC) are used, the input/output buffers must</td>
<td></td>
</tr>
<tr>
<td>meet this requirement</td>
<td></td>
</tr>
<tr>
<td>Used by CCOS to assign physical devices to the various files used</td>
<td></td>
</tr>
<tr>
<td>by the program</td>
<td></td>
</tr>
<tr>
<td>System macro-instruction used to define file control blocks</td>
<td></td>
</tr>
<tr>
<td>Pseudo-operation commonly used to define file designator words</td>
<td></td>
</tr>
<tr>
<td>Used to initialize files so that they may be properly accessed by</td>
<td></td>
</tr>
<tr>
<td>other File and Record Control (GFRC) functions</td>
<td></td>
</tr>
<tr>
<td>Used to obtain the next logical input record from a designated input</td>
<td></td>
</tr>
<tr>
<td>file</td>
<td></td>
</tr>
<tr>
<td>Used to move a logical record from working storage into an output</td>
<td></td>
</tr>
<tr>
<td>buffer</td>
<td></td>
</tr>
<tr>
<td>To initiate reading of a physical record with the user providing the</td>
<td></td>
</tr>
<tr>
<td>list of control words (DCW's)</td>
<td></td>
</tr>
</tbody>
</table>
The following statements are followed by a series of choices, only one of which is correct. Read each item carefully and mark the correct answer.

1. The command used to initialize the edit functions, such as those used when printing or punching is:
   - OPEN
   - IOEDIT
   - PRINT
   - PUNCH

2. The File and Record Control (GFRC) command with which the user can either allocate space within a buffer for a logical record and, if desired, move that logical record into the buffer is:
   - GET
   - PUT
   - GETBK
   - PUTSZ

3. The pseudo-operation used in conjunction with all File and Record Control (GFRC) commands is:
   - ERLK
   - IDRQ
   - SAVE
   - CALL

4. The File and Record Control (GFRC) command used to disconnect files on which no further activity is to be performed is:
   - CLOSE
   - PUTBK
   - RELSE
   - SETOUT

5. The File and Record Control (GFRC) command used to set a currently open file to an output file is:
   - CLOSE
   - PUT
   - SETIN
   - PUTBK
   - SETOUT

6. The File and Record Control (GFRC) command that may be used to obtain the next logical input record from a file is:
   - GET
   - PUT
   - CLOSE
   - RELSE
7. For disk and drum two types of files are provided:
   ___ Variable- and fixed-length
   ___ File Control Block and File Designator Word
   ___ Linked and Random
   ___ Letter and Nail

8. In the standard system format data blocks (physical records) may be variable in length up to a maximum of:
   ___ 320 bits
   ___ 321 words
   ___ 320 characters
   ___ 320 words

9. If in standard system format only one of the five media named below may use header and trailer labels. Indicate which.
   ___ Card Reader
   ___ Magnetic Tape
   ___ Paper Tape
   ___ Disk/Drum
   ___ Card Punch

10. The maximum block (physical record) size allowed with File and Record Control (GFRC) is:
    ___ 320 words
    ___ 1024 words
    ___ 4095 characters
    ___ 4095 words
1. Define the following terms:
   a. Physical Record:

   b. Logical Record:

   c. File Control Block:

   d. File Designator Word:

   e. Buffer:

   f. File:

2. What File and Record Control (GFRC) commands obtain information from
   the file designator words?

3. Why are there report codes and media codes in SYSOUT records?
4. Why must buffers that are used with logical records processing be one word larger than the largest physical record?

5. What are the types of records that can be handled by File and Record Control GFRC?

6. What are the characteristics of Standard System Format?

7. List the significant capabilities that are provided when Standard System Format is used.

8. What features does File and Record Control (GFRC) provide the user?
SECTION I
GMAP QUIZ 9

This section of the quiz consists of True-False items, each of which has a value of one (1) point. Enter your answer for each item on this sheet, making sure that you check the correct line for the corresponding test item.

1. The following coding will convert one binary word to seven (7) BCD characters.

   LDX1 0,DU
   LDA X
   RPT 6,1
   BCD PUB,1 (PUB is the start of the conversion table)
   STQ DATA
   BCD 0,1
   QLS 30
   STQ DATA+1

2. The following is a valid literal expression: =12H JUMP HOME.

3. STCA LOC,40 will affect bit positions 18-23.

4. STCA LOC,04 will affect bit positions 18-23.

5. HERE TRA **,5 will transfer to location HERE plus the value in X5.

6. LDX4 =1B18,DU will turn on bit zero in X4.

SECTION II

This section of the quiz consists of multiple choice items, each of which has a value of two (2) points. Each wrong answer will deduct two points from the total. Enter your answer in the A, B, C, or D column on this sheet, making sure that you check the correct line for the corresponding test item.
NOTE! There is only one (1) correct answer for each item in this section. Answer sheets which contain more than one answer per item will be considered to be wrong.

1. What is meant by the expression 1B17?
   A. A one in bit position seventeen.
   B. A type of airplane.
   C. A one followed by seventeen zero bits.
   C. A one followed by seventeen zeros.

2. In a floating-point number in memory the binary point of the mantissa is:
   A. Following bit 0
   B. Following bit 7
   C. Following bit 8
   D. Following bit 35

3. LDA  =.5B3,DU

   The above line of coding will cause which of the following octal configurations to be generated?
   A. 500000235003
   B. 400000235003
   C. 200000235003
   D. 000000235003

4. We wish to move fifty words from table A to table B. The two tables were established as follows:

   A  EBSS  50
   B  EBSS  50

   Which of the following coding examples (A or B) will do the job?

   A. LDX3  A,DU
       LDX4  B,DU
       RPD  25,2
       LDAQ  0,3
       STAQ  0,4

   B. LDX7  50,DU
            LDAQ  A-50,7
            STAQ  B-50,7
            SBX7  2,DU
            TNZ  *-3
5. \(=32.787\)
   
   This literal represents a/an:
   
   ___ A. Floating point number
   ___ B. Fixed point number
   ___ C. Integer
   ___ D. Octal literal

6. When using the DEC pseudo-op, what over-rides all other designations in determining whether the constant will be fixed or floating?
   
   ___ A. The letter B
   ___ B. The letter D
   ___ C. The letter E
   ___ D. A decimal point

7. A floating point number in memory has the following octal contents: 0002600000000. How many shift operations will be required to normalize this value?
   
   ___ A. 9
   ___ B. 6
   ___ C. 3
   ___ D. 1

SECTION III

This section of the quiz consists of completion type items, each of which has a value of three (3) points. Each wrong answer will deduct three points from the total.

NOTE! Each answer must be entirely correct to receive credit. No partial credit will be given (1 or 2 points) or items which are "almost correct.

All answers are brief, and should be entered in the space provided on the following sheet.
1. How many times will the instruction at X be executed?

LDX2 0,DU
RPT 14,1,TMI
X LDA Z,2
---
---
---
Z DEC 0,-1,-2,-3,-4,-5

2. The number 013461000000, is in core at location X. Show the octal contents of the E and A registers after execution of a FLD X.

3. Is the mantissa of the number in Item (Question) #2 positive or negative?
SECTION I

GMAP QUIZ 10

This section of the quiz consists of True-False items, each of which has a value of one (1) point. Enter your answer for each item on this sheet, making sure that you check the correct line for the corresponding test item.

1. If the letter Ø is coded in Column 7 of a source deck card, a NOP instruction may be inserted into the program by the assembler.

2. For the following coding the effective transfer address depends upon the contents of Index Register four (X4).

   TRA     TALLYX,I
   ---     ------
   ---     ------
   TALLYX  TALLYC  ADDRS,12,*4

3. The tally run-out indicator is not affected by using IT type modification.

4. The instruction: LDAQ Y will not execute if Y is an odd address.

5. STBA LOC,40 will affect bit positions 0-8.

6. STBA LOC,02 will affect bit positions 24-29.

SECTION II

This section of the quiz consists of multiple choice items, each of which has a value of two (2) points. Each wrong answer will deduct two points from the total. Enter your answer in the A, B, C, or D column on this sheet, making sure that you check the correct line for the corresponding test item.

NOTE! There is only one (1) correct answer for each item in this section. Answer sheets which contain more than one answer per item will be considered to be wrong!

1. The correct coding at the end of subroutine DØIT which will return us back to the normal return is:

   A. TRA  BETA,1
   B. TRA  2,1
   C. TRA  BETA+1
   D. TRA  TSXL+3

TSXL  DOIT
ARG  ALPHA
ARG  BETA
STA  BOOL

335
2. GOOD LDA ALPHA,*N
     ALPHA LDQ BETA,*5
     BETA CMPA STAR,DU

In the above program, the effective address of the LDA instruction is:

____ A. ALPHA
____ B. STAR+C(X5)
____ C. BETA+C(X5)
____ D. ALPHA+C(X5)

3. What is the effective address of the following coding sequence?

     LDA   W,*
     W --- X,6*
     X+C(X6) --- Y,7*
     Y+C(X7) --- Z

____ A. W
____ B. X+C(X6)
____ C. Y+C(X7)
____ D. Z

4. The instruction 000026735011B is contained in memory location BETA. Its representation on a symbolic coding form would be:

____ A. BETA ALS 22,1
____ B. BETA ALS 26,1
____ C. BETA ALR 22,1*
____ D. BETA ALR 26,1*

5. What is the maximum number of levels of indirection permitted?

____ A. Less than 63
____ B. 63
____ C. 64
____ D. Time limit consideration only.
6. LDQ PLACE,*

The above coding example employs which type of address modification?

____ A. IR
____ B. RI
____ C. IT
____ D. CI

SECTION III

This section of the quiz consists of completion type items, each of which has a value of three (3) points. Each wrong answer will deduct three points from the total.

NOTE! Each answer must be entirely correct to receive credit. No partial credit will be given (1) or 2 points) or items which are "almost" correct.

All answers are brief, and should be entered in the space provided on this sheet.

1. LDX3 0,1*

Prior to the execution of the above instruction, the contents of the various registers and memory cells are as follows:

(ALL ARE IN OCTAL)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>XI</td>
<td>000177</td>
</tr>
<tr>
<td>X3</td>
<td>002207</td>
</tr>
<tr>
<td>177</td>
<td>00140600000</td>
</tr>
<tr>
<td>contents</td>
<td>00111400000</td>
</tr>
<tr>
<td>of core</td>
<td>1406 00030700000</td>
</tr>
<tr>
<td>locations</td>
<td>1760 00001400000</td>
</tr>
<tr>
<td></td>
<td>2207 00176000000</td>
</tr>
</tbody>
</table>

What will be the octal contents of index register three after execution?

2. Referring to the information and conditions set forth in Item (Question) #1, what will be the octal contents of index register one be after execution?
3. **DUMMY** MACRO

\[
\begin{align*}
K &\quad \text{SET} & & 0 \\
&\quad \text{IDRP} & & \#2 \\
K &\quad \text{SET} & & K+1 \\
&\quad \text{IDRP} \\
&\quad \text{TRA} & & *+K \\
&\quad \text{ENDM} & & \text{DUMMY}
\end{align*}
\]

If we use the above MACRO with this calling sequence:

\[
\text{DUMMY ABLE,}(2,3,4,5,6,7),\text{BAKER}
\]

What would be the actual value of \( K \) when the transfer instruction is built by the assembler?
GMAP EXAM

SECTION ONE

This section of the exam consists of twenty True-False items, each of which has a value of one (1) point. Each wrong answer will deduct one point from the total of twenty possible points for this section. Enter your answer for each item on the answer sheet provided, making sure that you check the correct line for the corresponding test item.

1. The slave prefix area can be accessed by a program in the slave mode.

2. The slave service area can be accessed by a program in the slave mode.

3. The BLOCK pseudo-op may be used to assemble data and/or instructions into blank common.

4. The TZE instruction will transfer and turn off the zero indicator if it is on.

5. The following coding will convert one binary word to seven (7) BCD characters.

   \[
   \begin{align*}
   \text{LDX1} & \quad 0,\text{DUI} \\
   \text{LDA} & \quad X \\
   \text{RPT} & \quad 6,1 \\
   \text{BCD} & \quad \text{PUB,}1 \\
   \text{STQ} & \quad \text{DATA} \\
   \text{BCD} & \quad \text{PUB,}1 \\
   \text{QLS} & \quad 30 \\
   \text{STQ} & \quad \text{DATA+1}
   \end{align*}
   \]
   \(X\) is some binary number
   \(\text{PUB}\) is the start of the conversion table

6. The following is a valid literal expression: \(=12\text{H}\) JUMP HOME.

7. If the letter \(\varnothing\) is coded in Column 7 of a source deck card, a NOP instruction may be inserted into the program by the assembler.

8. For the following coding the effective transfer address depends upon the contents of Index Register four \((X4)\).

   \[
   \begin{align*}
   \text{TRA} & \quad \text{TALLYX,}1 \\
   \text{---} & \quad \text{--------} \\
   \text{---} & \quad \text{--------} \\
   \text{TALLYX} & \quad \text{TALLYC} \\
   \text{ADDRS,}12,\text{*4}
   \end{align*}
   \]

9. The tally run-out indicator is not affected by using IT type modification.

10. The instruction: LDAO Y will not execute if Y is an odd address.

11. The expression \((Y\text{-pair})\) means any two sequential memory locations that begin at an even address.
12. The contents of an index register may be incremented or decremented.

13. STCA LOC,40 will affect bit positions 18-23.

14. STCA LOC,04 will affect bit positions 18-23.

15. STBA LOC,40 will affect bit positions 0-8.

16. STBA LOC,02 will affect bit positions 24-29.

17. HERE TRA **,5 will transfer to location HERE plus the value in X5.

18. LDX 3,4 will load X3 with bits 0...17 of location 4.

19. LDX4 3,DL will load X4 with zeros.

20. LDX4 =1B18,DU will put a bit in X4.

Go on to SECTION TWO of the test at this time.
GMAP EXAM

SECTION TWO

This section of the exam consists of twenty-five multiple choice items, each of which has a value of two (2) points. Each wrong answer will deduct two points from the total of fifty possible points for this section. Enter your answer in the A, B, C, or D column on the answer sheet, making sure that you check the correct line for the corresponding test item.

NOTE! There is only one (1) correct answer for each item in this section. Answer sheets which contain more than one answer per item will be considered to be wrong.

21. Which of the following coding examples represents IR type modification?

A. ADR, N*
B. ADR, 2*
C. ADR, *
D. ADR, 2 N

22. What is meant by the expression =1B17?

A. a one in bit position seventeen.
B. a type of airplane.
C. a one followed by seventeen zero bits.
D. a one followed by seventeen zeros.

23. Given the following MACRO prototype:

MAC MACRO
LDA #1
ADA #2
STA #3
ENDM MAC

Assuming that all symbols have been defined, which of the following examples contains improper coding to use the above MACRO?

A. PLEASE MAC =20DL, =30DL, CAT
B. GO MAC (ABLE,4), CAT, DOG
C. GET MAC GOOD, BAD, INDIFF
D. HELP MAC (4), =45, UNABLE

24. At assembly time--

LDA =MLDX15, DU

A. will generate an entry in the literal pool.
B. The LDA is an instruction literal.
C. The LDA is a direct operand instruction.
D. Invalid coding.
25. In a floating-point number in memory the binary point of the mantissa is:
   A. following bit 0
   B. following bit 7
   C. following bit 8
   D. following bit 35

26. The following coding is used to go to a subroutine called "DIT" with the indicated normal return:

   TSX1 DIT
   ARG ALPHA
   ARG BETA
   STA DIT,0
   (this is the normal return)

   The correct coding at the end of subroutine DIT which will return us back to the normal return is:
   A. TRA BETA,1
   B. TRA 2,1
   C. TRA BETA+1
   D. TRA TSX1+3

27. LDA =MTRAIL,5

   The above coding is an example of:
   A. Direct upper (DU) address modification.
   B. An instruction literal.
   C. An instruction literal, but the ",5" is illegal.
   D. An illegal operand.

28. The decimal value of the following floating point number contained in a core dump is:
   7707000000000
   A. 5.4687
   B. 0.54687
   C. 0.054687
   D. 0.0054687

29. LDA =.5B3,DU

   The above line of coding will cause which of the following octal configurations to be generated?
   A. 500000235003
   B. 400000235003
   C. 200000235003
   D. 000000235003

30. In a floating-point literal, why would you use a D instead of an E or a decimal point?
   A. To indicate that this is a decimal literal.
   B. To cause a literal table dump if the program aborts.
   C. To obtain a double-precision number.
   D. To assemble the literal directly into the operand of the instruction.
31. GOOD LDA ALPHA, *N
ALPHA LDQ BETA, *S
BETA CMPA STAR, DU

In the above program, the effective address of the LDA instruction is:
A. ALPHA
B. STAR+C(X5)
C. BETA+C(X5)
D. ALPHA+C(X5)

32. What is the effective address of the following coding sequence?

\[
\begin{align*}
\text{LDA} & \quad W,* \\
W \quad --- & \quad X,6* \\
X+C(X6) \quad --- & \quad Y,7* \\
Y+C(X7) \quad --- & \quad Z
\end{align*}
\]
A. W
B. X+C(X6)
C. Y+C(X7)
D. Z

33. The instruction 0000267350118 is contained in memory location BETA. Its representation on a symbolic coding form would be:
A. BETA ALS 22,1
B. BETA ALS 26,1
C. BETA ALR 22,1*
D. BETA ALR 26,1*

34. Given the following coding example, what will be the contents of the A and Q registers after execution of the LDAQ instruction?

<table>
<thead>
<tr>
<th>000062</th>
<th>LDAQ</th>
<th>TABLE2</th>
</tr>
</thead>
<tbody>
<tr>
<td>000063</td>
<td>TRA</td>
<td>OUT</td>
</tr>
<tr>
<td>000064</td>
<td></td>
<td>DEC 5</td>
</tr>
<tr>
<td>000065</td>
<td>DEC 6</td>
<td></td>
</tr>
<tr>
<td>000066</td>
<td>DEC 7</td>
<td></td>
</tr>
<tr>
<td>000067</td>
<td>TABLE2 DEC 1</td>
<td></td>
</tr>
<tr>
<td>000070</td>
<td>DEC 2</td>
<td></td>
</tr>
<tr>
<td>000071</td>
<td>DEC 3</td>
<td></td>
</tr>
</tbody>
</table>

A. A = 6, Q = 7
B. A = 7, Q = 1
C. A = 1, Q = 2
D. A = 2, Q = 3

35. Indicate the coding which will store characters 2, 4, and 5 from the Q register to the same character positions of location X.
A. STCQ X,13
B. STCQ X,26
C. STCQ X,31
D. None of the above.
36. We wish to move fifty words from table A to table B. The two tables were established as follows:

A  EBSS  50  
B  EBSS  50  

Which of the following coding examples (A or B) will do the job?

A.  
LDX3  A,DU  
LDX4  B,DU  
RFD  25,2  
LDAQ  0,3  
STAQ  0,4  

B.  
LDX7  50,DU  
LDAQ  A-50,7  
STAQ  B-50,7  
SFX7  2,DU  
TNZ  *-3

37. TABLE OCT -43

The above line of coding will cause which of the following octal configurations to be generated?

A.  100000000043  
B.  777777777735  
C.  777777777743  
D.  400000000043

38. LDA 5,DU

In this instruction, the number five is:

A.  found in bits 18 - 35 of the instruction itself.  
B.  found in bits 0 - 17 of the instruction itself.  
C.  memory location five.  
D.  memory location five as modified by the DU directive.

39. LDA ,DU

The above instruction will cause which of the following octal contents to appear in bits 0 - 17 of the instruction?

A.  000023  
B.  000000  
C.  235003  
D.  235007

40. STX2 0,DL

What, if anything, is wrong with this instruction?

A. Nothing is wrong with this coding.  
B. The DL should be changed to a DU.  
C. The zero operand is illegal in slave mode.  
D. Direct type address modification is not allowed with a STX2.
41. What is the maximum number of levels of indirection permitted?
   A. Less than 63
   B. 63
   C. 64
   D. Time limit consideration only.

42. \(=32.7B7\)
   This literal represents a/an:
   A. floating point number
   B. fixed point number
   C. integer
   D. octal literal

43. When using the DEC pseudo-op, what over-rides all other designations
    in determining whether the constant will be fixed or floating?
   A. The letter B
   B. The letter D
   C. The letter E
   D. A decimal point

44. A floating point number in memory has the following octal contents:
    \(000260000000_8\). How many shift operations will be required to normalize
    this value?
   A. 9
   B. 6
   C. 3
   D. 1

45. LDQ PLACE, *
    The above coding example employs which type of address modification?
   A. IR
   B. RI
   C. IT
   D. CI

Go on to SECTION THREE of the test at this time
SECTION THREE

This section of the exam consists of ten completion type items, each of which has a value of three (3) points. Each wrong answer will deduct three points from the total of thirty possible points for this section.

NOTE! Each answer must be entirely correct to receive credit. No partial credit will be given (1 or 2 points) for items which are "almost" correct.

All answers are brief, and should be entered in the space provided on the answer sheet.

46. TSX1 30113

Prior to the execution of the above instruction, the IC contains the value 1122\text{\textsubscript{8}} and the 30113 is an octal number. What will the IC contain (in octal) after execution?

47. Referring to the information and conditions set forth in Item (Question) \#46, what will be the octal contents of index register one after execution?

48. How many times will the instruction at X be executed?

\begin{verbatim}
LDX2 0,DU
RPT 14,1,TMI
X   LDA Z,2
 ---
 ---
Z   DEC 0,-1,-2,-3,-4,-5
\end{verbatim}

49. The number 013461000000\text{\textsubscript{8}} is in core at location X. Show the octal contents of the E and A registers after execution of a FLD X.

50. Is the mantissa of the number in Item (Question) \#49 positive or negative?

51. LDX3 0,1*\text{\textsubscript{10}}

Prior to the execution of the above instruction, the contents of the various registers and memory cells are as follows:

(ALL ARE IN OCTAL)

\begin{align*}
\text{x1} & \quad 000177 \\
\text{x3} & \quad 002207 \\
\text{contents} & \quad 001406000000 \\
\text{of core} & \quad 001140000000 \\
\text{locations} & \quad 000307000000 \\
1760 & \quad 00000140000000 \\
2207 & \quad 001760000000
\end{align*}

What will be the octal contents of index register three after execution?

52. Referring to the information and conditions set forth in Item (Question) \#51, what will be the octal contents of index register one be after execution?
53. **DUMMY MACRO**

```
K
SET  0
IDRP  #2
K
SET  K+1
IDRP
TRA  *+K
ENDM  DUMMY
```

If we use the above MACRO with this calling sequence:

```
DUMMY  ABLE,(2,3,4,5,6,7),BAKER
```

What would be the actual value of K when the transfer instruction is built by the assembler:

54. Show the **binary** contents of a literal generated by: $=1.5B7$

55. Show the **octal** equivalent of the number generated in Item (Question) #54.

---

**BEFORE TURNING IN YOUR ANSWER SHEET:**

1. Be sure that your name and class number appear on the sheet.

2. Be sure that you have not forgotten to answer any items on the exam.

Please retain this exam question sheet for use during the review.

There are a total of 100 points available on this exam. It should be mentioned that your performance in this course, however, is not determined solely from the numerical grade achieved on this exam. The grade is used, along with several other factors, to contribute to a detailed personal evaluation of your work in this course.
<table>
<thead>
<tr>
<th>ITEM</th>
<th>TRUE</th>
<th>FALSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ITEM</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ITEM</th>
<th>ANSWER</th>
</tr>
</thead>
<tbody>
<tr>
<td>46</td>
<td>IC</td>
</tr>
<tr>
<td>47</td>
<td>X1</td>
</tr>
<tr>
<td>48</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>E</td>
</tr>
<tr>
<td>50</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>X3</td>
</tr>
<tr>
<td>52</td>
<td>X1</td>
</tr>
<tr>
<td>53</td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>(binary)</td>
</tr>
<tr>
<td>55</td>
<td>(octal)</td>
</tr>
</tbody>
</table>
The Other Computer Company:

Honeywell