ADVANCED COMPUTER SYSTEMS
APPLICATIONS, TECHNIQUES
AND CONCEPTS

June, 1968

Training Course
Prepared and Presented
by
AUERBACH Institute
## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction and Course Outline</td>
<td>1.1</td>
</tr>
<tr>
<td>Origins of Operating Systems and Multiprogramming</td>
<td>2.1</td>
</tr>
<tr>
<td>Multiprocessors</td>
<td>3.1</td>
</tr>
<tr>
<td>Transition to Third Generation Machines: System/360</td>
<td>4.1</td>
</tr>
<tr>
<td>Third Generation Operating Systems</td>
<td>5.1</td>
</tr>
<tr>
<td>Univac 1108 and Introduction to High Performance Machine</td>
<td>6.1</td>
</tr>
<tr>
<td>Realtime Considerations for Operating Systems: Univac 1108 Executive Systems</td>
<td>7.1</td>
</tr>
<tr>
<td>High Performance Hardware</td>
<td>8.1</td>
</tr>
<tr>
<td>High Performance Software</td>
<td>9.1</td>
</tr>
<tr>
<td>Stack Machines and Other Advanced System Concepts</td>
<td>10.1</td>
</tr>
<tr>
<td>Stack Machine Executives and Precision Considerations</td>
<td>11.1</td>
</tr>
<tr>
<td>Resource Allocation and Time Sharing</td>
<td>12.1</td>
</tr>
<tr>
<td>Systems Development for Time Sharing: GE 635 and OS 360/67</td>
<td>13.1</td>
</tr>
<tr>
<td>Software for Time Sharing -- MULTICS and TSS/360</td>
<td>14.1</td>
</tr>
<tr>
<td>Basic Concepts in Programming Languages</td>
<td>16.1</td>
</tr>
<tr>
<td>Structure of ALGOL</td>
<td>17.1</td>
</tr>
<tr>
<td>Data Management Environment</td>
<td>18.1</td>
</tr>
<tr>
<td>Structure of PLI</td>
<td>19.1</td>
</tr>
<tr>
<td>The Job Management Function</td>
<td>20.1</td>
</tr>
<tr>
<td>Assemblers, Symbol Tables and Macros</td>
<td>21.1</td>
</tr>
<tr>
<td>File System</td>
<td>22.1</td>
</tr>
<tr>
<td>Programming Systems</td>
<td>23.1</td>
</tr>
<tr>
<td>Data Management Technology</td>
<td>24.1</td>
</tr>
<tr>
<td>List Processing</td>
<td>25.1</td>
</tr>
<tr>
<td>Data Management Technology and Conversational Systems</td>
<td>26.1</td>
</tr>
<tr>
<td>String Manipulation Languages</td>
<td>27.1</td>
</tr>
<tr>
<td>Simulation Languages</td>
<td>28.1</td>
</tr>
<tr>
<td>Software Projection for the Near Future</td>
<td>29.1</td>
</tr>
<tr>
<td>Hardware Projection for the Near Future</td>
<td>30.1</td>
</tr>
<tr>
<td>Bibliography</td>
<td>A.1</td>
</tr>
</tbody>
</table>
- First and only time course will be offered
- Course was prepared under contract for NASA
- Instructors are not Auerbach employees

- Instructors
  - Jim Anderson
  - Beryl Bleckstein
OBJECTIVE OF FIRST HALF OF COURSE

- DESCRIBE THE ORIGINS AND DEVELOPMENTS OF SYSTEMS ARCHITECTURAL FEATURES PRESENT IN 3RD GENERATION COMPUTERS.

  - HARDWARE
    - MULTIPROCESSORS
    - MICROPROGRAMMING
    - INTERRUPT SYSTEMS
    - SCRATCH PAD
    - HIGH PERFORMANCE MACHINES
    - STACK MACHINES
    - TIME SHARING SYSTEMS

  - OPERATING SYSTEMS
    - MULTIPROGRAMMING
    - RESOURCE ALLOCATION
    - TIME-SHARING
    - REAL-TIME

PURPOSE: TO IDENTIFY IMPORTANT ORGANIZATIONAL CONCEPTS FOR COMPUTING SYSTEMS, AND THEIR AREA OF GREATEST APPLICABILITY.
INTRODUCTION AND COURSE OUTLINE

MAJOR INFLUENCES

- OPERATING EASE
- THRUPUT
- PROGRAMMER SERVICE
- LANGUAGE SUPPORT
TRADEOFFS

- MORE SERVICE
- LESS SPACE
- MORE EXECUTION TIME
- LESS PROGRAMMING TIME
- THE EGO QUESTION
INTRODUCTION AND COURSE OUTLINE

THE SYSTEM AS AN ENVIRONMENT

- USER HAS NO CHOICE
- FACILITIES PROVIDED:
  - PROGRAM INSERTION
  - DEBUGGING
  - LANGUAGE TRANSLATION
  - CORRECTION
- MORE SOPHISTICATION
  - I/O
  - LINKAGE
  - MULTIPLE LANGUAGE SUPPORT
A 3-PHASE OPERATING SYSTEM

1. SOURCE PROGRAM
2. SCAT
3. BINDER
4. PROG
5. EXECUTION
6. DEBUG
7. I/O
8. DATA
9. OUTTRAN
10. SNAPS
11. SNAP TRAN

PHASE 1
PHASE 2
PHASE 3
BINDING CONCEPT

- TIME WHEN PROGRAM IS ASSIGNED ACTUAL LOCATIONS IN MEMORY
- EARLY SYSTEMS – DURING CODING
- BY ASSEMBLER
- BY RELOCATION AND LINKAGE PROCESS
- BY SYSTEM, VIA COMPACTING
- DYNAMICALLY, BY PROCESS CALL
## Use and Definition Tables for a Program Unit

### Definition Table

<table>
<thead>
<tr>
<th>ENTRY SYMBOL</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### External Symbol #1

<table>
<thead>
<tr>
<th>USE 1 VALUE</th>
<th>USE 2 VALUE</th>
<th>...</th>
<th>USE n VALUE</th>
</tr>
</thead>
</table>

### External Symbol #2

<table>
<thead>
<tr>
<th>USE 1 VALUE</th>
<th>USE 2 VALUE</th>
<th>...</th>
<th>USE n VALUE</th>
</tr>
</thead>
</table>

### Program Text

- ENTRY 1
  - CALL EXTERNAL 3
  - CALL EXTERNAL 2
  - CALL EXTERNAL 1
INTRODUCTION AND COURSE OUTLINE

MAIN/SUB-PROGRAM ORGANIZATION

LINKAGE METHOD 1: DIRECT

PROGRAM UNIT "MAIN"

PROGRAM UNIT "TRIG"

PROGRAM UNIT "ZILCH"

PROGRAM UNIT "CRUD"
LINKAGE METHOD 2: TRANSFER VECTOR

PROGRAM

0
EXTERNAL SYMBOL 1

1
EXTERNAL SYMBOL 2

... 

n-2
EXTERNAL SYMBOL n-1

n-1
EXTERNAL SYMBOL n

n
OPR n-1

OPR 1

OPR 0

1.10
INTRODUCTION AND COURSE OUTLINE

LINKAGE METHOD 3: EXECUTION MAPPING

STORAGE MAPPING TABLE

EXT 1 LOCATION

EXT 1

CALL EXT 1

CALL EXT 1

CALL EXT 1

P1

P2

P3
FIELDS OF TYPICAL ASSEMBLER STATEMENT

- SYMBOLIC LOCATION NAME OR LABEL
- OPERATION
- OPERAND
- COMMENTS
- SERIAL IDENTIFICATION
RELOCATABLE SYMBOL RULES

REL: LABEL IN MACHINE ORDER OR LOCATION-DEFINING OPERATION
NONREL: LABEL IN NON-LOCATION-DEFINING OPERATION

REL + 5 → REL
REL + NONREL → REL
REL + REL → NONREL
REL * REL → NONREL

ALPHA = CLA + B (REL)
Z = EQU 7 (NONREL)
Y = EQU ALPHA (REL)
X = EQU ALPHA + 7 (REL)
W = EQU ALPHA * Z (NONREL)
## TEXT AND RELOCATION DATA

<table>
<thead>
<tr>
<th></th>
<th>R₁</th>
<th>R₂</th>
<th>R₃</th>
<th>R₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **PROGRAM TEXT WORDS**
- **RELOCATION DATA**
INTRODUCTION AND COURSE OUTLINE

TYPES OF STATEMENTS IN A TYPICAL ASSEMBLER

- MACHINE INSTRUCTIONS
- DATA DEFINING PSEUDO-OPERATIONS
- "BUILT-IN" SYSTEM MACROS
- ASSEMBLER CONTROL PSEUDO-OPERATIONS
- CONDITIONAL AND ASSIGNMENT OPERATIONS
- MACRO DECLARATIONS AND CALLS
OPERATION OF A SIMPLE TWO-PASS ASSEMBLER  
(a) PASS 1; (b) PASS 2

(a) INITIALIZE FOR PASS 1
   SLR = 0

   READ SYMBOLIC INSTRUCTION

   TEST FOR END OF PROGRAM
   NO

   TEST FOR LOCATION SYMBOL
   NO

   ENTER SYMBOL IN SYMBOL TABLE WITH CURRENT VALUE OF SLR

   INCREMENT SLR

   GO TO PASS 2
   YES

(b) INITIALIZE FOR PASS 2
   RESET SCAN TO FIRST RECORD
   SLR = 0

   READ SYMBOLIC INSTRUCTION

   END
   YES

   LOOK-UP OPERATION CODE

   LOOK-UP ADDRESS CODE

   ASSEMBLE AND STORE BINARY INSTRUCTION

   INCREMENT SLR

   NO
INTRODUCTION AND COURSE OUTLINE

DATA DEFINING PSEUDO-OPERATIONS:
- OCT
- DEC
- CHAR
- PREFIX CODES

BUILT-IN SYSTEM MACROS:
- CALL
- SAVE
- RETURN
- VARIOUS I/O OPERATIONS
- SUPERVISOR SERVICE REQUESTS

ASSEMBLER CONTROL PSEUDO-OPERATIONS:
- START
- END
- PRINT
- PUNCH
- ORG
- BSS
- USE
- ENTRY
- EXTERNAL
CONDITIONAL AND ASSIGNMENT STATEMENTS

\[
\begin{align*}
S & \quad \text{SET} \quad E \quad V(E) \quad \rightarrow \quad V(S) \\
S & \quad \text{SET} \quad 1 \quad 1 \quad \rightarrow \quad V(S) \\
S & \quad \text{SET} \quad S + 1 \quad V(S) + 1 \quad \rightarrow \quad V(S)
\end{align*}
\]

IF A, B, L

(IF \(V(A) = V(B)\) THEN SKIP ASSEMBLY TO LOCATION L)

IFF A, B

(IF \(V(A) = V(B)\), THEN SKIP COUNTER BY 2)
CALCULATION OF N FACTORIAL BY ASSEMBLER

N EQU
.
.
.
.
.
.
S SET 1
K SET 1
M IF S, N, L
S SET S + 1
K SET K * S
GO TO M
L CONTINUE
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
MACRO DEFINITION AND CALL

SUM  MACRO  A, B, C
LDA   A
ADD   B
STO   C
ENDM

ALPHA SUM ADDEND, AUGEND, TOTAL

ALPHA LDA ADDEND
ADD AUGEND
STO TOTAL

1.20
# INTRODUCTION AND COURSE OUTLINE

## ITERATIVE REPEAT FUNCTION

<table>
<thead>
<tr>
<th>SUM</th>
<th>MACRO</th>
<th>A, B, C</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>ADD</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>IRP</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>STO</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>IRP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENDM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ALPHA</th>
<th>SUM</th>
<th>X, Y, (Z1, Z2, Z3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ADD</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>STO</td>
<td>Z1</td>
<td></td>
</tr>
<tr>
<td>STO</td>
<td>Z2</td>
<td></td>
</tr>
<tr>
<td>STO</td>
<td>Z3</td>
<td></td>
</tr>
</tbody>
</table>

1.21
DEFINITION OF MULTIPROGRAMMING

THE TIME SHARING OF A CPU BY THE SEQUENTIAL OPERATION OF MULTIPLE PROGRAMS.
ORIGINS OF MULTIPROGRAMMING

- CPU TIME $< <$ I/O TIME

- VISIBLY SLOW EARLY MACHINES

- INTRODUCTION OF LARGER (E.G. 32K) MEMORIES
FUNCTION OF SIMPLE MULTIPROGRAMMING SUPERVISOR

- Decide the order of execution among resident jobs based on:
  
  Availability of data and facilities
  
  The priority of the job
  
  Relative priorities of other jobs
## Types of Multiprogramming

<table>
<thead>
<tr>
<th>Job Mix Memory Allocation</th>
<th>Fixed Content</th>
<th>Fixed Number</th>
<th>Variable Number and Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed Partitions</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Variable—Static</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Variable—Dynamic</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

2.5
FIXED PARTITION – FIXED CONTENT

- EARLIEST MULTIPROGRAMMING

- IN EFFECT COMBINED TWO PROGRAMS INTO ONE; PROGRAMS SHIFTED CONTROL BACK AND FORTH.

- CHOICE OF PROGRAMS CRITICAL – ONE 'COMPUTATIONAL,' ONE I/O BOUND

- NO INTERNAL SCHEDULING – COOPERATIVE CONTROL
FIXED PARTITION  FIXED NUMBER

- MODEL FOR PRESENT 360 DOS

- EARLY EMPHASIS ON MIX (E.G., COMPUTATIONAL AND I/O)

- IN PRINCIPLE ANY PROGRAM CAN BE RUN AS LONG AS IT FITS PARTITION

- USES EXECUTIVE TO SCHEDULE CPU TIME ON (POTENTIALLY) POSITION IN MEMORY I/O ACTIVITY PRIORITY

- MINIMUM USEFUL LEVEL OF MULTIPROGRAMMING
VARIABLE–STATIC FIXED NUMBER

- ALMOST COMPLETE — FIXED NUMBER OF PROGRAM ESTABLISHED TO FIX SIZE OF OP. SYSTEM TABLES

- SEQUENCING THROUGH RESIDENT PROGRAMS
  ROUND–ROBIN
  FIFO
  PRIORITY
  JOB LIST POSITION–DEPENDENT
  TIMER LIMITATIONS

- INTRODUCES MEMORY MANAGEMENT PROGRAMS
  COMPACTING FOR FREE SPACE
  ALLOCATION MADE AT LOAD TIME
  PERMITS QUEUEING JOBS ON SECONDARY STORAGE
VARIABLE—STATIC VARIABLE NUMBER AND CONTENT

- SIMILAR CAPABILITIES AS WITH FIXED NUMBER —
  MAY BE ABLE TO GET SOME FEW MORE PROGRAMS IN.

- REQUIRES MEMORY ALLOCATION FOR OPERATING SYSTEM
  AS WELL.
VARIABLE—DYNAMIC VARIABLE NUMBER AND CONTENT

- MODEL FOR MOST 'LARGE SCALE' MULTIPROGRAMMING SYSTEMS

- PERMITS RUN–TIME ALLOCATION OF MEMORY FOR HANDLING COMPLEX PROGRAM STRUCTURES

- PERMITS RUN–TIME COLLECTION AND BINDING OF PROGRAMS
  FORK
  JOIN
OTHER MULTIPROGRAMMING OPERATING SYSTEM ISSUES

- CONTROL INTERPRETERS

- RESOURCE ALLOCATION FOR QUEUED JOBS
  PERIPHERAL DEVICES
  MEMORY
  RESERVATION TECHNIQUES
CONTRIBUTIONS TO MACHINE ORGANIZATION

- HONEYWELL 800

- BASE REGISTER CONCEPT
THREE STAGES OF MULTIPROCESSOR DEVELOPMENT

1. HIGHER PERFORMANCE SYSTEMS THROUGH CONCURRENT PROCESSING

2. HIGH RELIABILITY SYSTEMS

3. IMPROVED PERFORMANCE AND SYSTEMS BALANCE
MULTIPROCESSOR
MULTIPROCESSOR

UNIVAC LARC

1

2 MEMORY

......

10

CU1

CU2

TAPE SYNCH

PRINTER SYNCH

CARD SYNCH

I/O PROCESSOR

DRUM SYNCH

PAGE RECORDER SYNCH

P. R.
MULTIPROCESSOR

TRW-400

(TOTAL 64)

(CENTRAL EXCHANGE)

(TOTAL 16)
MULTIPROCESSOR MODULAR MEMORY

0  4096  8192
4095  8191  12.288

COMPUTER I/O ACCESS BUSSES
MULTIPROCESSOR

TIME-SLOTTED BUS

CPU

I/O CHANNEL

CPU

I/O CHANNEL

MEMORY SUBSYSTEM

2-WAY DATA PATH
BANK SWITCHING (3600)

SELECTS MODULE

MODULE DESIGNATION

SELECTS WORD

WORD ADDRESS
ACCESS DISTRIBUTION ON LARC BUS

4 µS

<table>
<thead>
<tr>
<th>I/O</th>
<th>INST. OR OPERAND ACCESS</th>
<th>COMP 1 INST. ACCESS</th>
<th>COMP 2 OPERAND ACCESS</th>
<th>I/O P DISPACT. ACCESS</th>
<th>NOT USED</th>
<th>COMP 2 INST. ACCESS</th>
<th>COMP 1 OPERAND ACCESS</th>
<th>I/O DISP. ACCESS</th>
</tr>
</thead>
</table>
## CROSS BAR SWITCHED MEMORY

<table>
<thead>
<tr>
<th>MEMORY #1</th>
<th>MEMORY #2</th>
<th>MEMORY #3</th>
<th>MEMORY #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>CPU</td>
<td>CPU</td>
<td>CPU</td>
</tr>
<tr>
<td>CHANNEL</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.10
CROSS BAR SWITCH MEMORY (SHOWING UNIQUE CONNECTION TO EACH PROCESSOR BUS)

MEMORY UNIT #1  MEMORY UNIT #2  MEMORY UNIT #3  MEMORY UNIT #4

CPU

CPU

....

I/O

3.11
"DISTRIBUTED" CROSS BAR SWITCH

MEMORY SWITCH LOGIC
MEMORY SWITCH LOGIC
MEMORY SWITCH LOGIC
MEMORY SWITCH LOGIC

\( \ldots \)
CHANNEL SHARING MEMORY CIRCUITS OF CPU

- **ARITH. & CONTROL**
- **MEMORY ADDRESS REGISTER**
- **MEMORY CONTROL**
- **MAIN MEMORY**
- **MEMORY BUFFER REGISTER**
- **CHANNEL**
- **(INHIBIT)**

3.13
## REPRESENTATIVE MEMORY MODULE SIZES

<table>
<thead>
<tr>
<th>MACHINE</th>
<th>SIZE</th>
<th>MAXIMUM PERMITTED IN SYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1108</td>
<td>32K WORDS</td>
<td>8</td>
</tr>
<tr>
<td>360/65, 67</td>
<td>256K BYTES, (32K WORDS)</td>
<td>8</td>
</tr>
<tr>
<td>625/35/45</td>
<td>32K OR 64K</td>
<td>8 (4)</td>
</tr>
</tbody>
</table>
INDEPENDENT I/O CHANNEL

ARITHMETIC CONTROL

CHANEL

(INTERRUPTS)

(CONTROL)

DATA PATHS

MEMORY LOGIC

STORAGE
D825 CHANNEL ARRANGEMENT

MULTIPROCESSOR

M1  M2  M3  ...  M16

C1  ...  C3

IOC 1  IOC 1

IOC 2  IOC 2

IOC 2  IOC 10

I/O BUS.  I/O BUS.

PERIPHERAL EXCHANGE

PERIPHERAL DEVICES

3. 16
MULTIPROCESSOR

360/67 CHANNEL ARRANGEMENT
(SIMPLIFIED)

M_1  M_2  \ldots  M_U

P_1

P_2

C_{C_1}  C_{C_2}

C_{G_1}  C_{G_2}

TCU
SINGLE BASE REGISTER AND MEMORY ALLOCATION

BASE REGISTER

PROGRAM

DATA

STORAGE
SEPARATE PROGRAM AND DATA BASE REGISTERS

PROGRAM BASE REGISTER

DATA BASE REGISTER

DATA

PROGRAM

STORAGE
MULTIPROCESSOR

BASE REGISTER ADDRESSING

2174
BASE REGISTER

CUA 506

503
INSTS.

10 111 0101

*(506,
RELATIVE, 2680
ACTUAL)

STORAGE
360.RX INSTRUCTION

<table>
<thead>
<tr>
<th>OP CODE</th>
<th>R₁</th>
<th>X₂</th>
<th>B₂</th>
<th>D₂</th>
</tr>
</thead>
</table>

3.21
EVENT PROPAGATION IN AN OPERATING SYSTEM

CPU DISPATCHER \(\rightarrow\) USER PROGRAM (S) \(\Rightarrow\) I/O CONTROL

USER PROGRAM RETURN POINT, ID (CPU ID)

DATA REQUIRED, PLACE IN USER PROGRAM TO RESUME PROCESSING, (ID)
TYPICAL UNIPROCESSOR INTERRUPT IMPLEMENTATION

INTERRUPT

INTERRUPT TRANSFER VECTOR

IHR₁

IHR₂

IHR₃

...

IHR₄

STORAGE
MULTIPROCESSOR

INTERRUPT CLASSES ON 360

MACHINE CHECK (FAULT)
EXTERNAL
SUPERVISOR CALL
PROGRAM (FAULT)
I/O
360 CHANNEL STATUS WORD

<table>
<thead>
<tr>
<th>KEY</th>
<th>0000</th>
<th>COMMAND ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STATUS</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>47</td>
</tr>
</tbody>
</table>

- 32 ATTENTION
- 33 STATUS MODIFIER
- 34 CONTROL UNIT END
- 35 BUSY
- 36 CHANNEL END
- 37 DEVICE END
- 38 UNIT CHECK
- 39 UNIT EXCEPTION
- 40 PROGRAM-CONTROLLED INTERRUPT
- 41 INCORRECT LENGTH
- 42 PROGRAM CHECK
- 43 PROTECTION CHECK
- 44 CHANNEL DATA CHECK
- 45 CHANNEL CONTROL CHECK
- 46 INTERFACE CONTROL CHECK
- 47 CHAINING CHECK
METHODS OF PROGRAM & DATA PROTECTION

- BOUNDS REGISTERS
- STORAGE LOCKS
COMMON DATA IN A REAL-TIME APPLICATION

[Diagram showing common data storage and two programs (PGM #1 and PGM #2) accessing the common data]

STORAGE
TEST AND SET

- Cycle memory
- Fetch specified data, set byte in storage to all 1's
- Set condition code to 0 if leftmost bit is 0
  1 if leftmost bit is 1
FACTORS ENTERING INTO SYSTEM/360 DESIGN

- LARGE NUMBER OF EXISTING IBM INSTALLATIONS.

- COSTS OF MAINTAINING SEPARATE SOFTWARE SUPPORT FOR DIVERSE MACHINES.

- IMPACT OF HONEYWELL AND CDC.

- DESIRE TO CONSOLIDATE ALL LINES.

- BREAKDOWN OF SCIENTIFIC/BUSINESS DISTINCTION.

- GROWTH OF REAL-TIME APPLICATIONS.
TRANSITION TO THIRD GENERATION MACHINES: SYSTEM/360

SYSTEMS FAMILIES

704  1401  1604
   |       |      |
   709  1410  3600
      |       |
   7090 1440/7010
       |      |
   7094
       |      |
   7094II

*LARGELY COMPATIBLE AT MACHINE LANGUAGE LEVEL*
METHODS FOR CONVERTING BETWEEN MACHINES

- SIMULATION
- RE-COMPILATION
- LANGUAGE TRANSLATORS
- SUB-MACHINES
MACHINE SIMULATION AS CONVERSION AID

• Attempts to cope with conversion at machine-language level
• Complex program
• Usually cannot handle I/O directly
• Runs 1/100—1/1000 speed of machine being simulated
• Practical only if a very small number of infrequently run programs will be run on simulator
• Most frequently used as a design tool for new machines
• New machine(s) simulated on an older machine.
• Other difficulties
  — Word size compatibility
  — Special instructions (e.g., word mark handling on 1401)
  — Easy to overlook subtle machine features
  — Limits size of program that can run.
RECOMPILATION AS CONVERSION AID

- Assumes all of programs written in POL

- Original compiler can't have 'extensions' not present in second compiler

- Program does not take advantage of structure of original machine

- In general feasible only if lowest common denominator between two compilers was used

- POL's still not universally in use

- Slow development and acceptance of language standards

- Object programs run at target machine speed.
LANGUAGE TRANSLATOR AS CONVERSION AID

- WITH RECOMPIRATION, MOST SUCCESSFUL.

- CAN OPERATE AT MACHINE–LANGUAGE OR POL LEVEL
  
  — HONEYWELL LIBERATOR
  
  — BURROUGHS FORTRAN–TO–ALGOL TRANSLATOR.

- TO OPERATE AT MACHINE–LANGUAGE LEVEL, TARGET MACHINE MUST BE CLOSE REPLICA OF SOURCE MACHINE
  
  — HONEYWELL 200 LIKE IBM 1410.

- REQUIRES MANUAL FIXUP FOR I/O.

- WITH POL’S, CAN TRANSLATE TO EQUIVALENT LANGUAGE, ALTHOUGH FIXUP FOR MISSING FEATURES REQUIRED

  — BURROUGHS FORTRAN–TO–ALGOL SIMULATES SENSE SWITCH (LITE) OPERATORS IN FORTRAN

  — SIMSCRIPT TRANSLATES TO FORTRAN.

- OBJECT PROGRAMS RUN AT TARGET MACHINE SPEED.
SUBMACHINES AS CONVERSION AID

- WITHIN-FAMILIES, CAN BE USED.
- UNIVAC II OPERATED IN UNIVAC I MODE
- COMPATIBILITY SWITCH ON 709 TO RUN 704 PROGRAMS.
- DOESN'T HELP ACROSS MACHINE (MFGR.) LINES.
- MINIMUM COMPATIBILITY OF WORD SIZE, TAPE FORMATS.
- NOVEL, BUT NOT DONE EXCEPT WITH OLDER FAMILIES.
EMULATION – A SOLUTION TO CONVERSION PROBLEMS

- COMBINATION SIMULATION AND MICROPROGRAMMING.

- OBJECTIVES – TO EASE CONVERSION BY PROVIDING SIMULATION AT CLOSE TO ORIGINAL SPEEDS.

- PERMITS ORDERLY CHANGE OF MACHINES.

- WAS ALMOST MANDATORY WITH 360.

- MICROPROGRAMMING VALUABLE IN ITS OWN RIGHT.
MICROPROGRAMMING

- PROGRAMMING WITH ELEMENTARY MACHINE OPERATIONS.

- ELEMENTARY OPERATIONS
  - REGISTER TRANSFERS
  - ONE BIT SHIFT
  - MICROCODE BRANCHING.

- WILKES MACHINE.

- OBJECTIVES OF MICROPROGRAMMING PER SE
  - CUSTOM–TAILORED INSTRUCTION SETS
  - COST REDUCTION
  - CONTROL SYSTEM SIMPLIFICATION.
WILKES MICROPROGRAM CONTROL

ORDER REGISTER

CONTROL PULSES

REGISTER 11

CONTROL PULSES

REGISTER 1

MATRIX A

TO ARITHMETIC UNIT, REGISTER GATING ETC.

MATRIX B

FROM CONDITIONAL FLIP-FLOP

C. P.

IMPORTANT FEATURES OF WILKES DESIGN

- CONDITIONAL BRANCH.

- USE OF OP CODE AS ADDRESS OF FIRST MICROORDER.
OTHER IMPORTANT MICROPROGRAMMING DEVELOPMENTS

• MICRO SUBROUTINES.

• MICRO CONSTANTS.

• GROUPING FIELDS AND DECODING TO CONTROL PARTICULAR DATA PATHS.

• WRITABLE CONTROL STORAGE.

• TAILORED MACHINE LANGUAGE INSTRUCTION SETS.
USE OF MICROPROGRAMMING IN 360

- REDUCE CONTROL COSTS IN SMALLER MODELS.

- GIVE COMPREHENSIVE INSTRUCTION SETS ACROSS ALL MACHINE MODELS.

- PERMITS TAILORING FOR SPECIAL APPLICATIONS OR FOR VARIANTS ON BASIC LINE.

- READ–ONLY MEMORY (ROM) STORES MICRO–ORDERS.
EMULATOR DESIGN COMPONENTS

• DEDICATED ROM FOR EMULATOR (MAY BE SEPARATE ROM FOR MACHINE INSTRUCTIONS).

• SELECTION OF SPECIAL INSTRUCTIONS TO ADD TO BASIC (EMULATING MACHINE)
  
  — DIL
  
  — BRANCH IF.

• DETERMINE WHETHER FULL COMPATIBILITY OR SOME PROGRAMMED OPERATIONS
  
  — COST
  
  — COMPLEXITY
  
  — FREQUENCY OF OCCURRENCE.
LIMITATIONS OF MICROPROGRAMMING/EMULATOR APPROACH

- MICROPROGRAMMING ATTRACTIVE FOR COMPLEX AND/OR LARGE INSTRUCTION SETS.

- ROM TECHNIQUE FAST, INFLEXIBLE (TO USER).

- MAIN MEMORY (WRITEABLE CONTROL STORE) PERMITS GREATEST FLEXIBILITY TO USER.

- EMULATOR (MICROPROGRAM + PROGRAMS) FOR LARGER MACHINES.

- COMPATIBLE (THRU MICROPROGRAM) FOR SMALLER MACHINES.

- EMULATING MACHINE REGISTER STRUCTURE AND DATA PATHS MUST BE COMPATIBLE WITH TARGET MACHINE, GREATER DEVIATION, MORE COMPLEX AND DIFFICULT.

- NO RPQ OR OTHER NON-STANDARD FEATURES ON SOURCE MACHINE.
PRINCIPAL COMPONENTS OF 360 SYSTEM
INFORMATION STRUCTURE IN 360

- BYTE — (FAT CHARACTER)
- HALF WORD (2 BYTES)
- FULL WORD (4 BYTES)

- 7 BITS
- BYTE

- 15 HITS
- HALF WORD

- 31 BITS
- FULL WORD

- 63 BITS
- DOUBLE WORD
360 CPU STRUCTURE

PGM CTR

GENERAL REGISTERS

32
30
15
0

32
30
15
0

FLTG-PT REGISTERS

0
3
63
360 INSTRUCTION FORMATS

**REG. TO REGISTER**

<table>
<thead>
<tr>
<th>OP</th>
<th>R₁</th>
<th>R₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R₁)</td>
<td>&lt;OP&gt;</td>
<td>(R₂)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OP</th>
<th>R₁</th>
<th>X₂</th>
<th>B₂</th>
<th>D₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R₁)</td>
<td>&lt;OP&gt;</td>
<td>(D₂ ** X₂ B₂)</td>
<td>→</td>
<td>R₁</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OP</th>
<th>R₁</th>
<th>R₃</th>
<th>B₂</th>
<th>D₂</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>OP</th>
<th>I₂</th>
<th>B₁</th>
<th>D₁</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>OP</th>
<th>L₁</th>
<th>L₂</th>
<th>B₁</th>
<th>D₁</th>
<th>B₂</th>
<th>D₂</th>
</tr>
</thead>
</table>
360 TYPES OF OPERATIONS AND DATA FORMATS

HALF WORD

FULL WORD

FIXED POINT

PACKED DECIMAL

ZONED DECIMAL

ONE WORD

FLOATING POINT

DOUBLE WORD

FIXED LENGTH

VARIABLE

1-256 BYTES
### 360 STATEWORD — PSW

<table>
<thead>
<tr>
<th>SYSTEM MASK</th>
<th>KEY</th>
<th>AMWP</th>
<th>INTERRUPT CODE</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ILC</th>
<th>CC</th>
<th>PROGRAM MASK</th>
<th>INSTRUCTION ADDRESS</th>
</tr>
</thead>
</table>

- A — ASCII–8 MODE
- M — MACHINE CHECK MASK
- W — WAIT STATE
- P — PROBLEM STATE
- ILC — INST. LENGTH CODE
- CC — CONDITION CODE
ADDRESS FORMATION IN 360

**RX**

- **OP**
- **R₁**
- **X₂**
- **B₂**
- **D₂**

**RS**

- **OP**
- **R₁**
- **R₃**
- **B₂**
- **D₂**

**GENERAL REGISTERS**

**ADDRESS ADDER**

**EFFECTIVE ADDRESS**

**ADDRESS FORMATION IN 360**

(24)
TRANSITION TO THIRD GENERATION MACHINES: SYSTEM/360

360 INTERRUPT SYSTEM

PSW

CPU

INTERRUPT

'OLD' PSW STORAGE

PSW - X
PSW - S
PSW - P
PSW - M
PSW - I

'NEW' PSW STORAGE

PRIMARY STORAGE
TRANSITION TO THIRD GENERATION MACHINES: SYSTEM/360

360 CPU FEATURES FOR MULTIPROGRAMMING AND MULTIPROCESSING

- PROVIDES MULTIPLE BASE ADDRESSING (NOT IN ALL INSTRUCTIONS)

- COMPREHENSIVE INTERRUPT SYSTEM

- PROBLEM STATE/SUPERVISOR STATE

- NO INDIRECT ADDRESSING

- FIXED INTERRUPT RESPONSE LOCATIONS
### MEMORY SUBSYSTEM — 360

<table>
<thead>
<tr>
<th>MODEL</th>
<th>MINIMUM PRIMARY STORAGE</th>
<th>MAXIMUM PRIMARY STORAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>8,192</td>
<td>65,536</td>
</tr>
<tr>
<td>40</td>
<td>16,384</td>
<td>262,144</td>
</tr>
<tr>
<td>44</td>
<td>32,768</td>
<td>262,144</td>
</tr>
<tr>
<td>50</td>
<td>65,536</td>
<td>524,288</td>
</tr>
<tr>
<td>65</td>
<td>131,072</td>
<td>1,048,576</td>
</tr>
<tr>
<td>67</td>
<td>262,144</td>
<td>1,048,576</td>
</tr>
<tr>
<td>75</td>
<td>262,144</td>
<td>1,048,576</td>
</tr>
<tr>
<td>85</td>
<td>524,288</td>
<td>4,194,304</td>
</tr>
<tr>
<td>91</td>
<td>1,048,576</td>
<td>6,291,456</td>
</tr>
</tbody>
</table>

4.25
STORAGE PROTECTION – 360

- Process Memory Request
- Key: K₁, K₂, K₃
- Blocks: 2048
- Interrupt
ILLUSTRATION OF MEMORY INTERLACE

2-WAY INTERLACE

- FASTER OPERATION (ON AVERAGE) BY NOT HAVING TO WAIT FOR WRITE HALF-CYCLE
- FAILURE IN ONE MODULE EXCLUDES USE OF OTHER
LCS – SYSTEM IMPLICATIONS

- SIZE – 1M, 2M (UP TO 8M)

- SPEED – 8 μS

- AVAILABLE FOR MOD 50, 65, 75

- WHAT TO DO WITH IT
  - SYSTEM PROGRAMS RESIDENCE
  - FILE DIRECTORIES
  - OPERATING SYSTEM RESIDENCE
  - SWAPPING STORE
360 CHANNELS

- SELECTOR
  - 'BURST MODE' OPERATIONS
  - HIGH SPEED DEVICES

- MULTIPLEXOR
  - SLOWER DEVICES
  - SHARE MULTIPLEXOR LOGIC
  - USING SUBCHANNELS
CHANNEL COMMAND WORD – AN I/O PROGRAM

<table>
<thead>
<tr>
<th>COMMAND CODE</th>
<th>DATA ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FLAGS</th>
<th>COUNT.</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td></td>
</tr>
</tbody>
</table>

FLAGS:
- CHAIN DATA
- CHAIN COMMAND
- SUPPRESS LENGTH INDICATION
- SKIP
- PROGRAM CONTROLLED INTERRUPT
CHAINING

DATA AREA 1 (CCW 1)

DATA AREA 2 (CCW 2)

PRIMARY STORAGE

DA-1

DA-2

DA-4

DA-3

CCW 1

CCW 2

CCW 3

CCW 4
360 PROVISIONS FOR MULTISYSTEM OPERATION

- CPU COMMUNICATION
  - SHARED I/O – DISK
  - CHANNEL TO CHANNEL
  - SHARED STORAGE
  - CPU START SIGNAL (FROM ANOTHER CPU)

- INSTRUCTION AIDS
  - READ (WRITE) DIRECT
  - EXTERNAL INTERRUPT LINES
  - PERMANENT STORAGE RELOCATION AND ALTERNATE LOC. (PREFIX)
  - TEST AND SET
DIAGNOSTIC FACILITIES FOR 360

• 5 CLASSES OF INTERRUPTS
  - I/O
  - MACHINE CHECK
  - PROGRAM CHECK
  - SUPERVISOR CALL
  - EXTERNAL.

• PROGRAM CHECK ON
  - OPERATION EXCEPTION
  - PRIVILEGED—OPERATION EXCEPTION
  - EXECUTE EXCEPTION
  - PROTECTION EXCEPTION
  - ADDRESSING EXCEPTION
  - SPECIFICATION EXCEPTION
  - DATA EXCEPTION
  - FIXED POINT OVERFLOW
  - FIXED POINT DIVIDE
  - DECIMAL OVERFLOW
  - DECIMAL DIVIDE
  - EXPONENT OVERFLOW
  - EXPONENT UNDERFLOW
  - SIGNIFICANCE
  - FLOATING POINT DIVIDE
FEATURES OF RCA SPECTRA/70

- COPY OF 360 (PROBLEM MODE)

- HAS 4 PROCESSOR STATES
  1. PROBLEM (USER) STATE
  2. INTERRUPT RESPONSE STATE
  3. INTERRUPT CONTROL STATE
  4. MACHINE CONDITION STATE
## SPECTRA 70 PROCESSOR STATE REGISTERS

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>STATE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( P_1 )</td>
<td>( P_2 )</td>
</tr>
<tr>
<td>PROGRAM COUNTER</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>GENERAL REGISTERS</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>FLOATING POINT REGISTERS</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>INTERRUPT STATUS REGISTERS</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>INTERRUPT MASK REGISTERS</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
SUMMARY OF IMPORTANT CHARACTERISTICS OF S/70

- PROVIDES MULTICOMPUTER ARRANGEMENTS THROUGH DIRECT CONTROL TRUNK.

- EMULATORS FOR 301, 501 (RCA) 1401
  1410

- INTERNAL OPERATION LIKE 360
OPERATING SYSTEMS FOR 360

- BPS (BASIC PROGRAMMING SUPPORT)
- DOS (DISK)
- TOS (TAPE)
- OS (FULL)
- MFT (FULL WITH MULTIPROGRAMMING)
- MVT (FULL WITH VARIABLE TASKING)
THIRD GENERATION OPERATING SYSTEMS

THE DOS ENVIRONMENT

CONTROL PROGRAM

- SUPERVISOR
- JOB CONTROL
- IPL

PROCESSING PROGRAMS

- LANGUAGE TRANSLATORS
  - ASSEMBLER
  - COBOL
  - FORTRAN
  - PL/I
  - RPG

- SERVICE PROGRAMS
  - LINKAGE EDITOR
  - LIBRARIAN
  - SORT/MERGE
  - UTILITIES
  - AUTOTEST

- USER-WRITTEN
  - PROBLEM PROGRAMS

5.2
THIRD GENERATION OPERATING SYSTEMS

△ BACKGROUND
- CONTROL STREAM
- SEQUENTIAL
- USES JOB CONTROL
- NO OPERATOR INTERVENTION
- LOWEST PRIORITY

△ FOREGROUND
- NO CONTROL STREAM
- OPERATOR CONTROLLED
- USES INITIATORS
- HIGHEST PRIORITY

5.4
CREATION OF OVERLAY PHASE

MOD 1

CSECT1
CSECT2
CSECT3

MOD 2

CSECT4
CSECT5

PHASE PHNAME1,*
INCLUDE MOD1, (CSECT1, CSECT3)
INCLUDE MOD2, (CSECT5)

PHNAME 1

CSECT1
CSECT3
CSECT5
TWO PHASES FROM ONE OBJECT MODULE

MOD1

CSECT1

CSECT2

CSECT3

MOD2

CSECT4

CSECT5

PHASE PHNAME2, *
INCLUDE MOD1, (CSECT1, CSECT2)

PHASE PHNAME3, *
INCLUDE MOD1, (CSECT3)

PHNAME2

CSECT1

CSECT2

PHNAME3

CSECT3
USING SAME OBJECT MODULE TWICE

MOD1

CSECT1

CSECT2

CSECT3

MOD2

CSECT4

CSECT5

PHASE PHNAME4,*
INCLUDE MOD1, (CSECT1, CSECT2)

PHASE PHNAME5,*
INCLUDE MOD1, (CSECT2, CSECT3)
## Libraries

<table>
<thead>
<tr>
<th>CORE IMAGE DIRECTORY</th>
<th>RELOCATABLE DIRECTORY</th>
<th>SOURCE DIRECTORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHASE 1</td>
<td>C1</td>
<td>ASSEMBLER SUBLIB</td>
</tr>
<tr>
<td>PHASE 2</td>
<td>C2</td>
<td>COBOL SUBLIB</td>
</tr>
<tr>
<td>PHASE 3</td>
<td>C3</td>
<td></td>
</tr>
<tr>
<td>PHASE 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHASE 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHASE 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHASE 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHASE 4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.8
DIRECT LINKAGES

MAIN PROGRAM (A)  \rightarrow  FIRST-LEVEL SUBROUTINE (B)  \rightarrow  SECOND-LEVEL SUBROUTINE (C)

CALL  \rightarrow  SAVE  \rightarrow  SAVE

\rightarrow  CALL  \rightarrow  \rightarrow
call

\rightarrow  \rightarrow  \rightarrow
call

\rightarrow  \rightarrow  \rightarrow
call

\rightarrow  RETURN  \rightarrow  RETURN
THIRD GENERATION OPERATING SYSTEMS

GENERATION OF AN OVERLAY TREE STRUCTURE

PHASE PHASEA, ROOT
PHASE PHASEB,*
PHASE PHASEC,*
PHASE PHASED, PHASEC
PHASE PHASEE, PHASEB

ENTRY POINT

PHASEA

PHASEB

PHASEC

PHASED

PHASEE
JOB CONTROL EXAMPLE

// JOB EXAMPLE1

// OPTION LINK, LIST
// EXEC COBOL
  (COBOL Source Deck)
/*
// EXEC LINKEDT
// EXEC
  (Data for Object Program)
/*
/&

// JOB EXAMPLE2

// VOL SYS004,MASTER
// TPLAB 'label-information'
// EXEC PAYROLL
  (Data for Payroll Program)
/*
/&
OPERATING SYSTEM ELEMENTS

Control Program Elements

Job Management

Task Management

Data Management

Processing Program Elements

Languages

Service Programs

Application Programs

ALGOL

Assembler

COBOL

FORTRAN

PL/I

RPG

Data Set

Utilities

Independent

Utilities

Linkage

Editor

Sort/Merge

System

Utilities

TESTRAN

User

Written
PRODUCING A LOAD MODULE

User Input
Source Program + Translator Control Statements
Transcribed to Cards or Tape

Operating System Component

Output of Language Translator and Input to Linkage Editor
Linkage Editor Control Statements
Processed by a Language Translator Which Yields An Object Module

Operating System Component

Output of Linkage Editor
Processed by the Linkage Editor Which Yields A Load Module
## LOAD MODULE ATTRIBUTES

<table>
<thead>
<tr>
<th>Structure Type</th>
<th>Loaded All At One Time</th>
<th>Passes Control to Other Load Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Planned Overlay</td>
<td>No</td>
<td>No or Yes¹</td>
</tr>
<tr>
<td>Dynamic</td>
<td>Yes or No¹</td>
<td>Yes</td>
</tr>
</tbody>
</table>

¹A segment of a load module can dynamically call another load module.
SYSTEM LOGIC FLOW FOR A SIMPLE STRUCTURE

1. Find the Program SIMPLE
2. Allocate Space for it
3. Load the Entire Module
4. Supervise Execution
5. Give Control to Module
6. Terminate Task
STORAGE ALLOCATION FOR A PLANNED OVERLAY STRUCTURE

Storage Available to OVERLAY

Storage Occupied by Segment A (the Root Segment)

Storage When Segments A and B are Resident

Storage After Segment C Overlays Segment B
SYSTEM RESPONSE FOR A PLANNED OVERLAY STRUCTURE

1. Find the Program OVERLAY
2. Allocate Space for it
3. Load the Root Segment and Give It Control
4. Supervise Execution
5. Load Segment B
6. Supervise Execution
7. Overlay Segment B with Segment C
8. Supervise Execution
9. Supervise Execution
10. Terminate Task
THIRD GENERATION OPERATING SYSTEMS

DYNAMIC EXECUTION, ONE TASK PER JOB STEP

USER'S REQUEST

1. Find and Load DYNAMIC; Give It Control; Supervise Execution
2. Find, Load, and Give Control to Program A; Supervise Execution
3. Find, Load, and Give Control to Program B; Supervise Execution
4. Return Control to Program A at Instr Following LINK; Supervise Execution
5. Return Control to Program B; Supervise Execution
6. Supervise Execution
7. Supervise Execution
8. Terminate Job Step

SYSTEM RESPONSE

A. SAVE
B. RETURN
C. SAVE
D. RETURN

5. 19
DYNAMIC EXECUTION, MORE THAN ONE TASK PER JOB STEP

User Requests

1. Task A
   - Attach Task B
   - Wait Task B

2. Task B
   - Task B in Storage
     - Yes: Give Control to Task B, Supervise Execution
     - No: Find and Load Task B

System Response

1. Find, Load, and Give Control to Task A; Supervise Execution

2. Task B Needed
   - Task B in Storage
     - Yes: Give Control to Task B, Supervise Execution
     - No: Find and Load Task B

3. Task B Completed
   - Yes: Task B 1/O Completed; Give Control to Task A; Supervise Execution
   - No: Wait Until Task B Completed

4. Task BMust Wait for Completion of an I/O Event

5. Give Control to Task A; Supervise Execution

6. Task B 1/O Completed; Give Control to Task A; Supervise Execution

7. Task B Completed

8. Terminate Task B and Give Control to Task A

9. Terminate Job Step
THIRD GENERATION OPERATING SYSTEMS

REUSABILITY

- NON-REUSABLE
- SERIALLY REUSABLE
- REENTERABLE
DATA SETS, BLOCKS, AND RECORDS

DATA SET

BLOCK 1

| RECORD 1 |
| RECORD 2 |
| ... |
| RECORD n |

BLOCK 2

| ... |

BLOCK n
DESCRIBING A DATA SET

// DD

JOB STREAM

DATA SET LABEL

PROGRAM

DCB
DIRECT-ACCESS LABEL

The Volume Label

Volume Serial Number
Address of VTOC
Additional Labels, if Any

DSCB  DSCB  DSCB
for Data Set 29A4

The Volume Table of Contents - VTOC

Data Set 29A4
DATA SET RETRIEVAL THROUGH THE CATALOG

Catalog Volume

- Search for Input * Payroll * April Begins Here
- This Volume (21) Contains Data Set Input * Payroll * April
- Is Vol 21 Mounted?
  - Yes
  - Issue Mounting Message
  - Is Data Set on Tape or Direct Access?
    - Tape
    - Check Seq Number and Position: Seq No.

- Catalog (Major Entries)
- Index (Input *)
- Index (Input * Payroll *)

Check Seq Number and Position
PARTITIONED DATA SET

Data Set Address

Data Set Name

Partitioned Data Set

Directory

Members

Data Set A

Data Set B

Data Set C

Data Set D
INDEXED SEQUENTIAL DATA SET

Records Sorted on Key

Cylinder One

Cylinder Zero

Cylinder Index

Track Index

5.27
EXCHANGE BUFFERING -- SUBSTITUTE MODE

Original Buffer Assignments

- All Segments Assigned to Input Buffer
- Work Area
- All Segments Assigned to Output Buffer

After A "GET"

- This Segment Now Assigned to Work Area
- This Segment Now Assigned to Input Buffer

After A "PUT"

- This Segment Now Assigned to Output Buffer
- This Segment Now Assigned to Work Area
## ACCESS METHOD SUMMARY

<table>
<thead>
<tr>
<th>Organization</th>
<th>Sequential</th>
<th>Partitioned</th>
<th>Indexed Sequential</th>
<th>Direct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Primary</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>macro instructions*</td>
<td>GET, PUT,</td>
<td>READ, WRITE</td>
<td>SETL, GET, PUTX</td>
<td>READ,</td>
</tr>
<tr>
<td></td>
<td>PUTX</td>
<td>FIND, STOW</td>
<td>PUT</td>
<td>WRITE</td>
</tr>
<tr>
<td>Synchronization of</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>program with I/O</td>
<td>Automatic</td>
<td>CHECK</td>
<td>Automatic</td>
<td>WAIT</td>
</tr>
<tr>
<td>Record format</td>
<td>Logical F,V</td>
<td>Block (Part</td>
<td>Logical F,V</td>
<td>Block</td>
</tr>
<tr>
<td>transmitted</td>
<td>Block U</td>
<td>of member)</td>
<td>F,V</td>
<td>F,V,U</td>
</tr>
<tr>
<td>Buffer creation and</td>
<td>BUILD</td>
<td>BUILD</td>
<td>BUILD</td>
<td>BUILD</td>
</tr>
<tr>
<td>construction</td>
<td>GETPOOL</td>
<td>GETPOOL</td>
<td>GETPOOL</td>
<td>GETPOOL</td>
</tr>
<tr>
<td></td>
<td>Automatic</td>
<td>Automatic</td>
<td>Automatic</td>
<td>Automatic</td>
</tr>
<tr>
<td>Buffer technique</td>
<td>Automatic</td>
<td>GETBUF</td>
<td>Automatic, Simple</td>
<td>GETBUF ,</td>
</tr>
<tr>
<td></td>
<td>Simple</td>
<td>FREEBUF</td>
<td>Simple</td>
<td>FREEBUF</td>
</tr>
<tr>
<td></td>
<td>Exchange</td>
<td></td>
<td></td>
<td>Dynamic</td>
</tr>
<tr>
<td>Transmittal modes</td>
<td>Move, loc,</td>
<td>Move, loc,</td>
<td>Move, loc</td>
<td>Move,</td>
</tr>
<tr>
<td>(work area/buffer)</td>
<td>locate,</td>
<td>Locate</td>
<td></td>
<td>locate</td>
</tr>
</tbody>
</table>

*All macro instructions introduced in this table are defined in the publication IBM System/360 Operating System: Supervisor and Data Management Macro Instructions, Form C28-6647.
THIRD GENERATION OPERATING SYSTEMS

△ JOB MANAGEMENT FUNCTIONS
  • ANALYSIS OF INPUT STREAM (JCL)
  • ALLOCATION OF I/O DEVICES
  • OVERALL JOB SCHEDULING
  • TRANSCRIPTION OF INPUT/OUTPUT DATA
  • OPERATOR COMMUNICATIONS

△ FEATURES OF JOB CONTROL LANGUAGE
  • REFERENCING EXISTING STATEMENTS
  • DATA SET NAME RETRIEVING
  • OPTIMIZATION OF I/O
  • PASSING DATA SETS AMONG JOB STEPS
  • SHARING DATA SETS AMONG JOBS
TYPICAL JOB STATEMENTS

// DEMO1 JOB 62-7
// DEMO2 JOB (131-22, AZ6), TOM, MSGLEVEL = 1
// DEMO3 JOB 62-7, AL, PRTY = 13, REGION = 32K
// DEMO4 JOB 135, JOE, COND = (12, GT)
TYPICAL EXEC STATEMENTS

// STEP 1 EXEC PGM = MYCODE

// STEP 2 EXEC PGM = *.STEP6.MYDATA
// STEP 2 EXEC PGM = *.STEP7.PRSTEP2.YOURDATA
// STEP 2 EXEC PROC = CATPROC

// STEP 3 EXEC PGM = YOURCODE,COND = (17,EQ,STEP9)

// STEP 4 EXEC PGM = INTERP, TIME = (2,10),REGION = 64K
THIRD GENERATION OPERATING SYSTEMS

SOME TYPICAL DD STATEMENTS

// MYDATA DD SYSOUT = Z
// YOURDATA DD SYSOUT = 9, SPACE = (CYL,(7,1),RElse,ROUND)
// HISDATA DD UNIT = 180, DSNAME = HISSET, DISP = (CATLG, KEEP)
// HERDATA DD UNIT = 2311, DSNAME = HERSET, DISP = (CATLG)
// OURDATA DD DSNAME = OURSET, DISP = MOD, UNIT = TAPE, DEFER
// OLDDATA DD DSNAME = OLDSET, DISP = OLD, VOLUME = PRIVATE, RETAIN
// PASSDATA DD DSNAME = *.STEP3.HISDATA, DISP = (OLD, PASS)

5.33
A SEQUENTIAL SCHEDULING SYSTEM

1. Your programs, in the form of jobs or job steps defined through the job control language, may enter the system in the input stream from a card or tape device. Input data may be entered into the system with the control statements.

2. The reader/interpreter reads in the control statements for one job step.

3. The initiator/terminator allocates the required I/O devices, notifies the operator of volumes to be mounted (if any), and requests the task management programs to supervise execution of the named job step.

4. The task management programs turn control over to the first load module and supervise its execution.

5. The master scheduler accepts and takes action on commands.
A PRIORITY SCHEDULING SYSTEM

1. Your programs, defined as jobs or job steps by the job control language, enter the system through the input stream from a card or tape device.

2. The reader/interpreter reads in control statements for one or more jobs and places them, by priority, on the input work queue.

3. The job with the highest priority is selected for execution by the initiator/terminator.

4. The initiator/terminator turns your job step over to the task management programs, which supervise its execution.

5. The master scheduler accepts and takes action on commands.

6. Output is written (by job step priority) when the job has terminated and while other jobs are being processed.
THIRD GENERATION OPERATING SYSTEMS

RESOURCE QUEUES

Task Queues

D
PR = 3

C
PR = 4

B
PR = 10

A
PR = 12

Manager of CPU Time

C

A

Manager of Main Storage

B

A

Manager of I/O Channels

Queued Resource Requests

Queued Resource Requests
TOPICS FOR THIS SESSION

SCRATCH PAD MEMORY

COMPUTER NETWORKS

UNIVAC 1108

SYSTEM APPROACHES TO HIGH PERFORMANCE MACHINES.
DEFINITION OF SCRATCH PAD MEMORIES

SCRATCH—PAD MEMORIES:

SMALL, LOGIC—SPEED MATCHED MEMORIES
USED FOR REGISTERS AND/OR VERY HIGH SPEED WORKING STORAGE.
UNIVAC LARC

ACCUMULATOR OR INDEX REGISTER

PROCESSOR → 1 μSEC CORE

0

2500

4 μSEC CORE

0

PRIMARY STORAGE

(UP TO 40 MODULES)
D825 WITH THIN FILM REGISTER MEMORY

ARITHMETIC AND CONTROL

300 NS THIN FILM

(16 BIT WORDS)

4U SEC CORE

PRIMARY STORAGE

4096

6.4
### D825 Thin Film Registers

<table>
<thead>
<tr>
<th>Register Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Storage Register 1</td>
<td>(48)</td>
</tr>
<tr>
<td>Program Storage Register 2</td>
<td>(48)</td>
</tr>
<tr>
<td>Interrupt Program Register</td>
<td>(48)</td>
</tr>
<tr>
<td>Real-Time Clock</td>
<td>(24)</td>
</tr>
<tr>
<td>Repeat Count Register</td>
<td>(12)</td>
</tr>
<tr>
<td>Index Increment Register</td>
<td>(12)</td>
</tr>
<tr>
<td>Character Count Register</td>
<td>(12)</td>
</tr>
<tr>
<td>3 Repeat Increment Registers</td>
<td>(12 EA)</td>
</tr>
<tr>
<td>T-F C Register</td>
<td>(48)</td>
</tr>
<tr>
<td>Interrupt Storage Register</td>
<td>(48)</td>
</tr>
<tr>
<td>Subrouting Storage Register</td>
<td>(48)</td>
</tr>
<tr>
<td>Repeat Program Register</td>
<td>(64)</td>
</tr>
<tr>
<td>Interrupt Dump Register</td>
<td>(16)</td>
</tr>
<tr>
<td>Power Failure Dump Register</td>
<td>(32)</td>
</tr>
<tr>
<td>Program Count Register</td>
<td>(16)</td>
</tr>
<tr>
<td>Base Program Register</td>
<td>(16)</td>
</tr>
<tr>
<td>Base Address Register</td>
<td>(16)</td>
</tr>
<tr>
<td>Subroutine Base Address Register</td>
<td>(16)</td>
</tr>
<tr>
<td>Interrupt Base Address Register</td>
<td>(16)</td>
</tr>
<tr>
<td>15 Index Registers</td>
<td>(16 EA)</td>
</tr>
<tr>
<td>15 Limit Registers</td>
<td>(16 EA)</td>
</tr>
</tbody>
</table>
360 GENERAL REGISTERS

ACCUMULATORS, BASE OR INDEX REGISTERS

0 32

0 64

15

3
SPECTRA 70/35 SCRATCH-PAD MEMORY

GENERAL PURPOSE AND FLOATING POINT REGISTERS, PROGRAM COUNTER, INTERRUPT STATUS AND MASK REGISTERS.

ADDRESSABLE* PORTION

SUBCHANNEL REGISTERS FOR MULTIPLEXOR CHANNEL

NON-ADDRESSABLE

*IMPLEMENTED AS A SEPARATE MEMORY ON 70/45, 70/55.
STORE AND (IDXQ, PRTQ) ASSOCIATIVE MEMORY

INST. PROCESSOR

INST. EXEC. QUEUE

ARITHMETIC AND CONTROL

STACK EXTENSION

INST. LOOK AHEAD (12 WORDS)

REQUEST FOR DATA

SIMPLIFIED DIAGRAM B8500 AND PROCESSOR ILLUSTRATING PRINCIPAL SCRATCH-PAD MEMORIES

DATA/INSTS. FROM MEMORIES MODULES
LOOK-ASIDE MEMORY

ASSOCIATIVE STORAGE

PRIMARY STORAGE

DATA PART

ADDRESS PART

MAR

MBR

CPU
SUMMARY OF SCRATCH–PAD CHARACTERISTICS

● PRIMARY FUNCTIONS
  CLOSE-IN STORAGE MATCHED TO LOGIC SPEEDS, INEXPENSIVE IMPLEMENTATION OF CONTROL Registers, MASK REAL SPEED OF PRIMARY STORAGE

● POTENTIAL PROBLEMS
  CONTENTS OF SCRATCH–PAD BECOMES PART OF THE STATE OF AN ACTIVE PROCESS

● SOLUTIONS
  ASSOCIATIVE STORE
  MULTIPLE SCRATCH–PAD
TYPES OF COMPUTER NETWORKS

- DEDICATED
  - COMMUNICATIONS SWITCH
  - RESERVATION SYSTEMS
  - AIR DEFENSE SYSTEMS

- LOAD-SHARING
  - REMOTE COMPUTING
COMMUNICATIONS SWITCHING SYSTEM

ST+ FWD SWITCH

TRUNK LINES

ST+ FWD SWITCH

6.12
FUNCTIONS OF CONCENTRATOR

- It's a computer
- Speed matching
- Buffer for economical transmission to switch
- Local distribution

TO STORE AND FORWARD SWITCH

Diagram:

- C
- S S S
FUNCTIONS OF SWITCH

MULTIPLE-ADDRESS ROUTING
STORE AND FORWARD NETWORK PROBLEMS

• RELIABILITY
  
  MULTIPLE COMMUNICATIONS PATHS
  MULTIPROCESSOR OR MULTICOMPUTER ELEMENTS
  TRANSMISSION CONTROL
  CHECKING
  DISTRIBUTED CONTROL

• LONG TERM STORAGE
  
  MULTIPLE ADDRESS MESSAGES
  STATION LOGS

• EFFICIENT PROCESSING
  
  INDEPENDENT I/O (COMMUNICATIONS) CHANNELS

• PEAK LOADS
  
  SUFFICIENT SECONDARY STORAGE FOR BUFFERING
  DISC/DRUM
  TAPE
AIR-DEFENSE NETWORK
LOAD SHARING NETWORK

SATELLITE CENTERS

COMMUNICATIONS LINES

CENTRAL SITE
TYPES OF LOAD-SHARING

• REMOTE JOB PROCESSING
  SATELLITES ACCEPT DATA AND CONTROL INFORMATION
  CENTER QUEUES JOB FOR EXECUTION
  CENTER RETURNS RESULTS TO SATELLITE
  SATELLITE PRINTS RESULT
  GENERALIZATION OF DCS CONCEPT

• ACTIVE SATELLITES
  SMALL JOBS PERFORMED IN SATELLITE
  LARGE (FOR SATELLITE) JOBS PERFORMED REMOTELY

• FULL SHARING
  2 OR MORE CENTERS
  ALL JOBS DONE AT CENTER
  OVERLOAD AT ONE CENTER TRANSMITTED TO ANOTHER
SOME LOAD SHARING PROBLEMS

EQUIPMENT AND CONFIGURATION COMPATIBILITY

PROGRAM AND DATA LOCATION

AUTONOMOUS CENTERS

ALL THE COMMUNICATIONS PROBLEMS
UNIVAC 1108 AND INTRODUCTION TO
HIGH PERFORMANCE MACHINES

UNIVAC 1108 SIMPLIFIED MULTIPROCESSOR CONFIGURATION

BANK 1

<table>
<thead>
<tr>
<th>ODD</th>
<th>EVEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMA</td>
<td>MMA</td>
</tr>
</tbody>
</table>

65K

BANK 2

<table>
<thead>
<tr>
<th>ODD</th>
<th>EVEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMA</td>
<td>MMA</td>
</tr>
</tbody>
</table>

65K

BANK 3

<table>
<thead>
<tr>
<th>ODD</th>
<th>EVEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMA</td>
<td>MMA</td>
</tr>
</tbody>
</table>

65K

BANK 4

<table>
<thead>
<tr>
<th>ODD</th>
<th>EVEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMA</td>
<td>MMA</td>
</tr>
</tbody>
</table>

65K

CPU

MPA

I/OC

MPA

I/OC

MPA

MPA

DEV. CONT.

DEV. CONT.

DEV. CONT.

6.20
OVERLAPPED FETCH IN UNIVAC 1108

BANK 1

0

ODD

EVEN

OPERAND

MEMORY CONTROL LOGIC

MEMORY CONTROL LOGIC

32K

BANK 2

0

ODD

EVEN

NEXT INST.

MEMORY CONTROL LOGIC

MEMORY CONTROL LOGIC

32K

PROCESSOR
ADDRESSING AND STORAGE PROTECTION – UNIVAC 1108 – SIMPLIFIED

- BI
  - \( \mu + BI \)
  - \( \mu + BE + XM \)

- BD
  - \( \mu + BD \)
  - \( \mu + BD + XM \)

- \( \mu \)
  - \( U + XM = U \)
  - BSIU

- INST. LIMITS TEST
  - LOWER
  - UPPER

- DATA LIMITS TEST
  - LOWER
  - UPPER

- PRIMARY STORAGE REGISTERS

- REAL ADDRESS SPACE

- USER ADDRESS SPACE
1108 AS A MULTIPROCESSOR

- DESIGNED AS A UNIPROCESSOR
- MULTIPROCESSING CONNECTIONS THROUGH ADAPTORS
  MMA
  MPA
- FULL 1107 COMPATIBILITY
- GUARD MODE ≡ USER STATE (MODE)
- SEPARATE PROGRAM AND DATA AREA BOUNDS REGISTERS
- I/O OPERATES WITHOUT STORAGE PROTECT FEATURE
- ADDITIONAL MODULE FOR MULTIPROCESSOR SYSTEMS – AVAILABILITY CONTROL UNIT
PROBLEMS IN ATTAINING HIGH PERFORMANCE SYSTEMS

- EXTREME MISMATCH BETWEEN SPEED OF LOGIC AND PRIMARY STORAGE
- MISMATCH BETWEEN PRIMARY AND SECONDARY STORAGE
- SERIAL REPRESENTATION OF PROGRAMS
OVERLAP

INFORMATION AVAILABLE AT CPU

SIMPLE MACHINE

OVERLAPPED MACHINE (DATA AND INSTRUCTIONS RESIDING IN ALTERNATE BANKS OR MODULES)
LOOK-AHEAD

MEMORY CYCLE

I_A  I_B  OAD_B  D_B  X_B  I_C  I_D

OAD_A  D_A  X_A

6.27
PIPELINE

MEMORY CYCLE

A + B

MEMORY CYCLE

D_A D_B
I_C I_D

MEMORY CYCLE

X_A X_B
D_C D_D
I_E I_F

MEMORY CYCLE

X_C X_D...
D_E D_F
I_G I_H
FUNCTIONAL OUTLINE PIPELINE MACHINE

INST. QUEUE

DATA QUEUE

ARITHMETIC LOGIC

RESULT QUEUE

PRIMARY STORAGE
OTHER TECHNIQUES TO REDUCE LOGIC–MEMORY SPEED MISMATCH

• LOOKASIDE

• SCRATCHPADS
TECHNIQUES FOR REDUCING PRIMARY–SECONDARY STORAGE SPEED MISMATCH

- MULTIPLE CHANNELS
- HEAD PER TRACK DISC UNIT
- SECTOR QUEUES
DISC-SECTOR QUEUEING FUNCTIONAL DIAGRAM

DISC ACCESS QUEUE

<table>
<thead>
<tr>
<th>TR/D</th>
<th>OP</th>
<th>TR/D</th>
<th>OP</th>
</tr>
</thead>
<tbody>
<tr>
<td>TR/D</td>
<td>OP</td>
<td>TR/D</td>
<td>OP</td>
</tr>
<tr>
<td>TR/D</td>
<td>OP</td>
<td>TR/D</td>
<td>OP</td>
</tr>
<tr>
<td>TR/D</td>
<td>OP</td>
<td>TR/D</td>
<td>OP</td>
</tr>
<tr>
<td>TR/D</td>
<td>OP</td>
<td>TR/D</td>
<td>OP</td>
</tr>
</tbody>
</table>

SECTOR

1
2
3
4
5
6
7

DISC CONTROL

READ-WRITE SIGNALS

HEADS

6.32
SOURCE OF PARALLELISM IN PROGRAMS

- INDEPENDENT OPERATIONS
  STATEMENT LEVEL
  ARITHMETIC EXPRESSION LEVEL
- PARALLEL LOOPS
- OVERLAPPED LOOPS
(1) \[ A = B \]

(2) \[ C = A + 1 \]

(3) \[ D = B + 2 \]

(4) \[ B = B + 1 \]

INDEPENDENT STATEMENTS, 1, 3

2, 4

6.34
EXPRESSION PARALLELISM

EXPRESSION: \((B + C)/(E - F) + D - (R + P)/Q - M) + R\)

EXPRESSION TREE

ALL OPERATIONS AT SAME LEVEL ARE INDEPENDENT AND CAN BE EXECUTED IN PARALLEL
PARALLEL LOOP

\[ R = 5 \]

DO 10 I = 1, 10

\[ M = I + R \]

\[ A(I) = B(I) + M \]

10 CONTINUE

ITERATION 1
\[ M = 1 + R \]
\[ A(1) = B(1) + M \]

ITERATION 2
\[ M = 2 + R \]
\[ A(2) = B(2) + M \]

ITERATION 3
\[ M = 3 + R \]
\[ A(3) = B(3) + M \]
PARALLEL LOOP CHARACTERISTICS

- SAME OPERATION(S) APPLIED TO DIFFERENT DATA

- INDEX SET DETERMINES DATA IN A REGULAR MANNER

- PERMITS BULK EXECUTION OF PROGRAMS
REAL-TIME CONSIDERATIONS FOR OPERATING SYSTEMS
UNIVAC 1108 EXECUTIVE SYSTEM

▲ CAPABILITIES

• BATCH PROCESSING
• DEMAND REMOTE
• REAL-TIME COMMUNICATIONS

▲ FEATURES

• PROGRAM PROTECTION
  — MEMORY
  — RESERVED OPERATIONS
• MASS STORAGE UTILIZATION
• ELABORATE PROGRAM FILE SYSTEM
• CONTROL STATEMENTS MAY BE CATALOGUED
• MULTIPLE VERSIONS

▲ LANGUAGES

• FORTRAN
• COBOL
• ASSEMBLY
• ALGOL
• CONVERSATIONAL FORTRAN
BASIC CONCEPTS AND DEFINITIONS

- ACTIVITY
- BATCH
- COLLECTION
- FILES
  - GRANULES
  - PACKETS
- RUN
- TASK
- SWAPPING
- PRIVILEGED INSTRUCTIONS
SYSTEM COMPONENTS

- SUPERVISOR
- EXECUTIVE REQUESTS
- SYMBIONTS
- I/O HANDLERS
- OPERATOR COMMUNICATIONS
- FILE CONTROL
- DATA HANDLING
- FILE UTILITIES
- AUXILIARY PROCESSORS
  - COLLECTOR
  - PROCEDURE DEFINITION
  - LANGUAGE PROCESSORS
  - PROCESSOR INTERFACE ROUTINES
- DIAGNOSTIC SYSTEM
  - SNAPSHOTS
  - POST-MORTEM
- SYSTEM GENERATION
- UTILITY ROUTINES
REAL-TIME CONSIDERATIONS FOR OPERATING SYSTEMS
UNIVAC 1108 EXECUTIVE SYSTEM

▶ STATEMENT FORMAT

@ [LABEL] : COMMAND [OPTIONS] SPEC. LIST COMMENTS

▶ STATEMENT TYPES

- ORGANIZATIONAL
- I/O SPECS
- PROCESSOR CALLS
- PROGRAM EXECUTION
- CONDITIONAL
ORGANIZATIONAL STATEMENTS

@ RUN        APPEARS AT THE BEGINNING OF EACH RUN. PROVIDES ACCOUNTING AND IDENTIFICATION INFORMATION.

@ FIN        APPEARS AT THE END OF EACH RUN.

@ LOG        PLACES USER SPECIFIED INFORMATION IN THE SYSTEM LOG.

@ MSG        PLACES A MESSAGE ON THE CENTRAL–SITE CONSOLE TYPEWRITER.

@ HDG        USED TO PLACE A HEADING LINE ON PRINT OUTPUT.

@ ADD        USED TO DYNAMICALLY EXPAND THE RUN STREAM.

@ START      USED TO SCHEDULE THE EXECUTION OF AN INDEPENDENT RUN.

@ SYM        USED TO SCHEDULE NON–STANDARD SYMBIONT ACTION.

@ COL        USED TO SPECIFY VARIOUS FORMS OF INPUT.

@ CKPT       USED TO ESTABLISH A CHECKPOINT DUMP THAT MAY BE USED FOR RESTART AT SOME FUTURE TIME.

@ RSTRT      USED TO RESTART A RUN AT SOME PREVIOUSLY TAKEN CHECKPOINT.
INPUT/OUTPUT SPECIFICATION STATEMENTS

@ ASG  USED TO ASSIGN A PARTICULAR INPUT/OUTPUT DEVICE OR MASS STORAGE FILE TO A RUN. THERE ARE FOUR TYPES OF @ ASG STATEMENTS:
   FASTRAND
   TAPE
   DRUM
   ARBITRARY DEVICE
   ALSO USED TO CATALOGUE FILES.

@ MODE  USED TO CHANGE THE MODE SETTINGS (DENSITY, PARITY, ETC.) OF A TAPE FILE.

@ CAT  CATALOGUES FASTRAND FORMATTED OR EXISTING TAPE FILES.

@ FREE  USED TO DEASSIGN A FILE AND ITS INPUT/OUTPUT DEVICE OR MASS STORAGE AREA.

@ USE  USED TO SET UP A CORRESPONDENCE BETWEEN INTERNAL AND EXTERNAL FILE NAMES.

@ ELT  INSERTS OR UPDATES A PROGRAM–FILE ELEMENT FROM THE CONTROL STREAM.

@ DATA  USED TO INTRODUCE OR UPDATE A DATA FILE FROM THE CONTROL STREAM.

@ END  USED TO TERMINATE A DATA FILE.

@ FILE  USED TO CAUSE THE DIRECT CREATION OF A FILE CONTAINING DATA TAKEN FROM THE CONTROL STREAM.

@ ENDF  USED TO TERMINATE THE DATA THAT follows THE @FILE STATEMENT.

@ QUAL  USED TO DEFINE A STANDARD FILE NAME QUALIFIER.
PROGRAM EXECUTION STATEMENTS

@ MAP  USED TO CALL THE COLLECTOR AND PREPARE AN ABSOLUTE ELEMENT.

@ XQT  USED TO INITIATE THE EXECUTION OF A PROGRAM.

@ EOF  USED TO SEPARATE DATA WITHIN THE CONTROL STREAM.

@ PMD  USED TO TAKE EDITED POST-MORTEM DUMPS OF THE PROGRAM JUST EXECUTED.
PROCESSOR CALL STATEMENTS

@ PROCESSOR  USED TO EXECUTE A PROCESSOR (@COB FOR COBOL COMPILER, @FOR FOR FORTRAN, @ASM FOR ASSEMBLER, ETC.)
CONDITIONAL STATEMENTS

@ LABEL: Used to attach a label to an existing control statement.

@ SETC Places a value in the 'condition' word.

@ JUMP Used to branch control within the control stream.

@ TEST Used to test the 'condition' word in the course of deciding the effective control stream.
BATCH PROCESSING

- SIMPLE FORTRAN LOAD-AND-GO EXAMPLE:

  @ RUN AK4,888,OPTICS,5,75
  @ ASG,T ATMOS,T,A341
  @ FOR
  ....
  ....
  @ FORTRAN SOURCE
  ....
  XQT
  ....
  ....
  DATA
  ....
  @ PMD
  @ FIN

- A MORE COMPLEX EXAMPLE:

  @ RUN AL5,888,OPTICS,10
  @ ASG,T ATMOS,T,A341
  @ ASG SPEC,F SPECIAL FILE
  @ FOR PROGS. MURK(15) , PROGS. MURK/ABER
  .........
  CORRECTIONS TO CREATE MURK/ABER FROM MURK (15)
  @ MAP
  IN PROGS. MURK/ABER
  @ XQT
  @ SYM PRNT,SPEC
  @ FIN
DEMAND PROCESSING EXAMPLE

- USER SIGN-ON:
  - U1108 T/S 1
  - READY
  - # RUN XYZ,311202,DEMO
  - # ASG,C PF,F/5
  - # ASM,I PF,ODDEVEN
  - ASM 1/1/67

(Terminal identified with WRU.)
(The system is ready for first input.)
(The run begins with RUNID, ACCOUNT, and PROJECT number to identify the user.)
(A 5 TRACK FILE 'DEMO PF' IS ASSIGNED, TO BE CATALOGUED AT THE END OF RUN.)
(Start assembly of element called 'ODDEVEN'.)
(The assembler is ready to accept input.)
DEMAND PROCESSING EXAMPLE

ASSEMBLY LANGUAGE PROGRAM:

REGNAM

(A PROC TO DEFINE REGISTER NAMES IS CALLED FROM THE SYSTEM LIBRARY.)

P FORM 12,6,18
ST * P$PRINT (P 5,4,STMSG)
R$EAD (+ EXIT$,INPUT)

L A1, INPUT?

L,S1 A1, INPUT
L A0, (P 1'4'ODD)
JB A1,ST+1
L A0, (P 1'4'EVEN)

J ST+1
INPUT RES 14
STMSG 'TYPE A SINGLE NUMBER.'
ODD 'IT'S ODD; TRY ANOTHER.'
EVEN 'IT'S EVEN; TRY ANOTHER.'
END ST
DEMAND PROCESSING EXAMPLE

EXECUTION OF PROGRAM AND SIGN-OFF

ASM COMPLETE $0 000043

(THE ASSEMBLY IS FINISHED. PROGRAM IS 043 WORDS LONG.) (REQUEST EXECUTION.)

# XQT,N

TYPE A SINGLE NUMBER.
1 IT'S ODD; TRY ANOTHER.
4 IT'S EVEN; TRY ANOTHER.
A IT'S EVEN; TRY ANOTHER.
# FIN

(SMART PROGRAM—.)
(THAT'S ENOUGH.)

27/3/67 0945

RUNID: XYZ ACCOUNT: 311202 PROJECT: DEMO
TIME: 0000.02 IN: 00023 OUT: 00000 PAGES: 0001

(END OF TRANSMISSION REQUEST TO QUIT THE LINE.)

LINE RELEASED (LAST WORDS FROM SYSTEM.)
SUPERVISOR COMPONENTS

- RESIDENT ROUTINES
  - INTERRUPT SUPERVISOR.
  - CPU DISPATCHER.
  - INPUT/OUTPUT CONTROL.
  - DEVICE HANDLERS FOR TAPE, FASTRAND, COMMUNICATIONS
    SUB-SYSTEMS, ETC. (RECOVERY SEQUENCES ARE TRANSIENT).
  - DRUM HANDLER, INCLUDING RECOVERY SEQUENCES.
  - DYNAMIC_ALLOCATOR.
  - CORE CONTENTS CONTROL.
  - EXECUTIVE REQUEST SUPERVISOR.
  - REAL-TIME CLOCK AND DAY CLOCK ROUTINES.
  - BLOCK_BUFFERING_PACKAGE.
  - TASK AND SEGMENT_LOADER.
  - CONSOLE CONTROL.
  - BASIC QUEUEING_PACKAGE AND QUEUE AREA.
  - READS AND PRINTS.
  - LOGGING CONTROL.
  - ERROR INTERRUPT SUPERVISOR.
  - CORE PARITY RECOVERY ROUTINE.
  - POWER-LOSS CONTROL ROUTINE.
SUPERVISOR COMPONENTS

TRANSIENT ROUTINES

- CONTROL STATEMENT INTERPRETER.
- COARSE SCHEDULER.
- DEMAND CONTROL
- FACILITIES INVENTORY.
- SECONDARY FASTRAND SPACE ASSIGNMENT.
- COMMUNICATIONS INTERFACE ROUTINES.
- CLT DIAL-UP AND AUTOMATIC-ANSWER
- SYMBIONT PROBE ROUTINES.
- MISCELLANEOUS DEVICE HANDLERS (PAPER TAPE, ETC.).
- SYMBIONTS.
- CONSOLE HANDLER.
- LOGGING AND ACCOUNTING.
- I/O ERROR RECOVERY SEQUENCES FOR TAPE, FASTRAND, ETC.
- TAPE LABEL CHECKING.
- ABSOLUTE DUMP ROUTINE.
COARSE SCHEDULER

► BATCH PROCESSING
  • RUN QUEUE
  • STATEMENT QUEUE
    - WAIT FOR FACILITIES
    - BEING PROCESSED BY C. S.
    - IN CORE QUEUE
    - WAITING FOR OPERATOR
  • CORE QUEUE
    - ACTIVE
    - SUSPENDED
    - READY

► DEMAND PROCESSING
  • RUN
  • STATEMENT
  • CORE–SWAP QUEUE
    - ACTIVE
    - SWAPPED–OUT
    - READY
    - INPUT–WAIT
DYNAMIC ALLOCATOR

- CORE ALLOCATION
  - USES CORE CONTENT CONTROL (C. C. C)

- TIME ALLOCATION
  - DISPATCHER
  - PRIORITIES
    - REAL–TIME
    - CRITICAL DEADLINE
    - DEMAND
    - BATCH

- PROGRAM STATES
  - TERMINATED
  - SUSPENDED FOR HIGHER PRIORITY
  - WAITING FOR COMPLETION OF EXTERNAL EVENT
  - INPUT–WAIT
  - ACTIVE

7.17
DATA FLOW IN THE SUPERVISOR

RUN QUEUE

STATEMENT QUEUE

C. S.

CORE QUEUE

CORE–SWAP QUEUE

P. C. T

D. A.

SWITCH LIST

C. C. C.

CORE MAP

DISP

TIME MAP

7.18
**THE SWITCH LIST**

- **N-LEVEL, MULTIPLE ENTRY (L = 0, 1, 2, ..., N)**
- **INITIAL LEVEL = 0**
- **LEVEL L HAS PRIORITY OVER LEVEL L+1**
- **WITHIN LEVEL, CDU TIME PRIORITIES ARE EQUAL**
- **PROGRAM LOSES CONTROL BY VOLUNTARY OR INVOLUNTARY ACTION**
  - **THE TIME-LIMIT QUANTUM Q:**
    - \( T_L = 2^L \)
    - \( A = \text{ALLOCATION FACTOR BY D.A.} \)
    - \( F = \text{PRIORITY FACTOR} \)
    - \( Q = A \cdot (1 + P/F) \cdot T_L \)
  - **IF Q IS EXCEEDED, L + 1 \rightarrow L FOR THAT TASK**

- **SWITCH LIST FUNCTIONS FOR DA:**
  - **ENTER (INITIAL L FOR A TASK)**
  - **SET (ALTERS VALUE OF A FOR A TASK)**
  - **MOVE (ALTERS VALUE OF L FOR A TASK)**
  - **MOVE 1 (ALTERS VALUE OF L FOR ALL TASKS OF GIVEN TYPE)**
  - **MOVE 2 (INCREMENT OR DECREASES L FOR ALL TASKS OF GIVEN TYPE)**

7.19
DISPATCHER

- CPU GIVEN TO HIGHEST PRIORITY
- FULL LEVEL--CYCLE MUST BE COMPLETED
- DISPATCHER USES SWITCH LIST FOR:
  - ENTRY POINT
  - RUN ID.
  - STATEWORD
  - ACTIVITY MARK
  - MEMORY LOCKOUTS
  - RUNNING TIME
  - P. C. T. ADDRESS POINTER
FILE CONTROL SYSTEM

FUNCTIONS

- DIRECTORY MAINTENANCE
- MASS STORAGE ALLOCATION
- INTERFACE WORKER PROGRAMS AND DEVICE HANDLERS
- PROTECTION
**COLLECTOR EXAMPLE**

<table>
<thead>
<tr>
<th>FILEA ELEMENTS NAME/VERSION</th>
<th>REFERENCES OUTSIDE OF FILEA REQUIRED FILE, NAME/VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>FILEA, A1, B1, F1</td>
</tr>
<tr>
<td>A1/A</td>
<td></td>
</tr>
<tr>
<td>A2/A</td>
<td>LIB1, SIN/X</td>
</tr>
<tr>
<td>A3/A</td>
<td>LIB2, COS/X</td>
</tr>
<tr>
<td>B1/B</td>
<td>LIB1, SQRT/X</td>
</tr>
<tr>
<td>B2/B</td>
<td></td>
</tr>
<tr>
<td>B3/B</td>
<td></td>
</tr>
<tr>
<td>C1/C</td>
<td>LIB1, SQRT/X</td>
</tr>
<tr>
<td>C2/C</td>
<td></td>
</tr>
<tr>
<td>D1/D</td>
<td>LIB2, CAT/Y</td>
</tr>
<tr>
<td>D2/D</td>
<td></td>
</tr>
<tr>
<td>E1/E</td>
<td>LIB2, CAT/Y</td>
</tr>
<tr>
<td>E2/E</td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td></td>
</tr>
<tr>
<td>F2</td>
<td></td>
</tr>
<tr>
<td>G1/G</td>
<td>LIB1, SIN/X</td>
</tr>
<tr>
<td>G2/G</td>
<td>LIB2, COS/X</td>
</tr>
<tr>
<td>G3/G</td>
<td></td>
</tr>
</tbody>
</table>

A PARTICULAR COLLECTION SETUP FOR SEGMENTING A PROGRAM FROM THIS FILE MIGHT BE AS FOLLOWS:

```plaintext
MAP, L , X
SEG MAIN
IN FILEA, MAIN
SEG A*, (MAIN)
IN FILEA, A1/A, A2/A, A3/A
SEG B*, (A)
IN FILEA, B1/B, B2/B, B3/B
SEG C*, B
IN FILEA, C1/C, C2/C
SEG D*, (B, C)
IN FILEA, D1/D, D2/D
SEG E*, D
IN FILEA, E1/E, E2/E
DSEG F*, (D, G)
IN FILEA, F1, F2
SEG G*, (MAIN)
IN FILEA, G1/G, G2/G
LIB LIB1, LIB2
@ XQT
```

7.22
REAL-TIME CONSIDERATIONS FOR OPERATING SYSTEMS
UNIVAC 1108 EXECUTIVE SYSTEM

STORAGE MAP

INSTRUCTION AREA MEMORY MAP

<table>
<thead>
<tr>
<th>01000</th>
<th>CAT</th>
<th>B1-B2-B3---</th>
<th>D1-D2-----</th>
<th>K</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>COS</td>
<td>SQRT</td>
<td>-A1-A2-A3</td>
<td>E1-E2-----</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-MAIN-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>G1-G2</td>
<td></td>
<td>F1-F2------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DATA AREA MEMORY MAP

<table>
<thead>
<tr>
<th>N</th>
<th>CAT</th>
<th>B1-B2-B3---</th>
<th>D1-D2-----</th>
<th>E1-E2-----</th>
<th>O</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SQRT</td>
<td>-A1-A2-A3</td>
<td></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>ILDS$</td>
<td>-G1-G2-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LT-BC--MAIN-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MAIN (A1, B1, F1)

<table>
<thead>
<tr>
<th>G1 (SIN)</th>
<th>A1</th>
<th>CAT</th>
<th>SQRT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2 (SIN)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A3 (COS)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1 (SQRT)</td>
<td></td>
<td>B1</td>
<td>SQRT</td>
</tr>
<tr>
<td>C2</td>
<td></td>
<td>B2</td>
<td></td>
</tr>
<tr>
<td>G2 (COS)</td>
<td>D1</td>
<td>E1</td>
<td>CAT</td>
</tr>
<tr>
<td>D2</td>
<td></td>
<td>E2</td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.23
CONVERSATIONAL FORTRAN

- SERVICE LANGUAGE
  - PROGRAM ENVIRONMENT STATEMENTS
  - EXECUTION CONTROL
  - STATEMENT MODIFICATION
  - DISPLAY
  - TEST FUNCTIONS
    - TRACE (REPORT VALUE CHANGES)
    - TRAP (REPORT ALL TRANSFERS)
    - TRAIL (REPORT ALL EXTERNAL PROCEDURE CALLS)
    - DUMP
    - LIMIT (REPORT VALUE OUTSIDE LIMITS)
    - KEYIN (ALLOW CONSOLE CONTROL)
    - EX (IMMEDIATE, BUT NOT PERMANENT)
    - EXR (IMMEDIATE AND PERMANENT)
    - OFF

7.24
CONVERSATIONAL FORTRAN

@CFOR
+NOTE
101.
CONVERSATIONAL FORTRAN IN EFFECT
READY
@EX
READY
Z = SQRT (CONSTANT)
Z = VALUE
READY
Y = SIN (CONSTANT)
Y = VALUE
READY
R = SIN (CONSTANT)
R = VALUE
READY
@OFF (EX)
101.
.
.
.

@CFOR
+NOTE
101.
CONVERSATIONAL FORTRAN IN EFFECT
READY
@ACTIVITY TEST
READY
READ (2, 20), A, B, C
READY
10 A = B + C
READY
@UPDATE
*
READY
-101, 101
* 101.
READY
READ (2, 20), B, C
* 101. 1
READY
@OFF (UPDATE)
READY
@TRACE A
103.
READY
R = B/A + C
104.
READY
.
.
.

@CFOR
+NOTE
101.
CONVERSATIONAL FORTRAN IN EFFECT
READY
@ACTIVITY EXAMPLE
READY
@TRACE A, B, C
READY
READ (2, 20), A, B, C
READY
D = A − B + C
READY
A = D+C/A
READY
B = A−D
READY
20 FORMAT (F8. 3)
105.
READY
@BEGIN
-101.
READY (INPUT VALUES ENTERED FOR A, B AND C)
+TRC
103.
A = VALUE
+TRC
104.
B = VALUE
106.
READY

7.25
CONVERSATIONAL FORTRAN

@ CFOR
+NOTE CONVERSATIONAL FORTRAN IN EFFECT
101. READY DIMENSION A(100)
102. READY @ EXR
102. READY 20 FORMAT (F8.3)
103. READY READ (2, 20), B, C
-103. READY (INPUT VALUES ENTERED FOR B AND C)
104. READY D = 20
105. READY E = 20 - B - C
106. READY E = VALUE
107. READY E = 20 - C - B
107. READY E = VALUE
107. READY EX E = B/C + 19.9
107. READY E = VALUE
107. READY @ UPDATE
* READY -102
* 102.1 READY READ (2, 50), (A(I;)
* 102.2 READY , I = 1, 100)
* 102.2 READY @ OFF (UPDATE)
107. READY A(2) = B - C
108. READY A(3) = - A(2)
109. READY
### CONVERSATIONAL FORTRAN

@ CFOR

+ NOTE

<table>
<thead>
<tr>
<th>CONVERSATIONAL FORTRAN IN EFFECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>101. READY @ EXR</td>
</tr>
<tr>
<td>101. READY</td>
</tr>
<tr>
<td>102. READY</td>
</tr>
<tr>
<td>-102. READY</td>
</tr>
<tr>
<td>103. READY</td>
</tr>
<tr>
<td>104. READY</td>
</tr>
<tr>
<td>105. READY</td>
</tr>
<tr>
<td>106. READY</td>
</tr>
<tr>
<td>* 103. READY</td>
</tr>
<tr>
<td>* 103. 1 READY</td>
</tr>
</tbody>
</table>

STATEMENT AT 104. REQUIRES A LABEL

<table>
<thead>
<tr>
<th>READY</th>
</tr>
</thead>
<tbody>
<tr>
<td>104.</td>
</tr>
<tr>
<td>105.</td>
</tr>
<tr>
<td>106.</td>
</tr>
<tr>
<td>* 103.</td>
</tr>
<tr>
<td>* 103.1</td>
</tr>
</tbody>
</table>

IF (J-K) 5, 7, 9
IF (L) 4, 5, 9
L = L - 1

FORMAT (13)
READ (2, 10), J, K, L
INPUT VALUES ENTERED FOR J, K AND L
IF (J-K) 5, 7, 9
L = L - 1
IF (L) 4, 5, 9
L = VALUE

FORMAT (13)
READ (2, 10), J, K, L
IF (K-J) 5, 7, 9
L = L - 1

7.27
CONVERSATIONAL FORTRAN

@ CFOR
+ NOTE CONVERSATIONAL FORTRAN IN Effect

101. READY @ LIMIT A,GT,20
101. READY @ EXR
102. READY 20 FORMAT (F8,3)
103. READY READ (2,20), A,I
103. READY (INPUT VALUES ENTERED FOR A AND I)
104. READY @ OFF (EXR)
104. READY READ (2,20), (B(J), ; J=1,1)
105. READY 3 IF (B(I)) 5,15,4
106. READY 4 A = B(I) + SIN(B(I))
107. READY 5 A = B(I) * A
108. READY I I - 1
109. READY GO TO 3
110. READY 15 A = COS (B(I))
111. READY @ BEGIN 104
101. READY DIMENSION B (100)
104. READY (INPUT VALUES ENTERED FOR B-ARRAY)
+LMT A,GT,20 110. A = 21,75

READY
•
•
TOPICS COVERED THIS SESSION

- HIGH PERFORMANCE MACHINES
  - 6600
  - 360/9X, 360/85
  - B8500
  - ILLIAC IV
FUNCTIONAL ORGANIZATION—CDC 6600

12 CHANNELS

MULTIPLEXED PERIPHERAL AND CONTROL PROCESSOR

4K MEMORYS, 1 FOR EACH VIRTUAL P&C PROCESSOR

OPERATING REGISTERS

ADD, MUL, MUL, DIV, LONG ADD, SHIFT, BOOLEAN, INCREMENT, INCREMENT
PROGRAM INITIATION - 6600

PRIMARY STORAGE

N:  
1
16

P AND C PROCESSOR

6X00
CONTROL REGISTERS

8.3
HIGH PERFORMANCE HARDWARE

CDC-6400

1 CHANNELS — 12

MULTIPLEXED PERIPHERAL AND CONTROL PROCESSOR

1 2 . . . 32

INTEGRATED ARITH. AND CONTROL UNIT

1 2 . . . 10

4K MEMORY, 1 FOR EACH VIRTUAL P&C PROCESSOR
HIGH PERFORMANCE HARDWARE

CDC-6500

CHANNELS

MUX P&CP

INTEGRATED ARITH. AND CONTROL UNIT

INTEGRATED ARITH. AND CONTROL UNIT

1 2 10

1 2 32
HIGH PERFORMANCE HARDWARE

360/91 FUNCTIONAL DIAGRAM

HIGH PERFORMANCE PRIMARY STORAGE

.75 μs 256K

(INTERLEAVED 8 OR 16 WAYS)

MAIN STORAGE CONTROL ELEMENT

EXTENDED PRIMARY STORAGE

.75 μs 256K

(16 WAY INTERLEAVE)

PERIPHERAL STORAGE CONTROL ELEMENT (PSCE)

INST BUFFERS
STORG BUFFERS
OPER-AND BUFFERS

GEN.PURP. REGISTERS

FIXED POINT EXECUTION

FLOATING POINT EXECUTION

FLOATING POINT REGISTERS

CHANNELS AND I/O

8.6
MODULO 8 INSTRUCTION STACK - 360/91

SAMPLE POINTER SETTINGS

0
1
2
3
4
5
6
7

UB
LB
IR
AOC

OP REGISTER

8.7
ELEMENTS OF 360/91 CONTRIBUTING TO SPEED

- MULTIPLE INTERLEAVED HIGH SPEED STORAGE

- STORAGE ACCESS BUFFERING

- INSTRUCTION BUFFERING
  
  INST FETCH LOOKAHEAD
  SHORT LOOP EXECUTION

- OPERAND FETCH AND STORE BUFFERING

- MULTIPLE ARITHMETIC EXECUTION ELEMENTS
HIGH PERFORMANCE HARDWARE

360/85 TWO LEVEL STORAGE SYSTEM

CPU

LOCAL STORE

80 ns

BACKING STORE

80 ns
OBJECTIVES OF 360/85 2-LEVEL STORE SYSTEM

- GENERALIZATION OF LOOK-ASIDE MEMORY
- 'PAGE' CONCEPT APPLIED FOR INCREASED PERFORMANCE
- AMORTIZE ACTUAL ACCESS TIME OVER SEVERAL WORDS
HIGH PERFORMANCE HARDWARE

360/85 BUFFER MEMORY LOGIC

24 BIT ADDRESS

<table>
<thead>
<tr>
<th>BLOCK ID</th>
<th>B-D</th>
</tr>
</thead>
</table>

ASSOCIATIVE MEMORY

BLOCK ID (16)  BFMD (8)

MATCH

NO MATCH

PRIMARY STORAGE

BUFFER MEMORY

TO CPU
SUMMARY OF MODEL 85 CHARACTERISTICS

- MODEL 85 EMBODIES 'LOOK ASIDE' CONCEPT

- IMPLEMENTATION SIMILAR TO PAGING IN 360/67 (TO BE DISCUSSED)

- WITH THE PARAMETERS CHOSEN, DATA OR INSTRUCTIONS FOUND IN BUFFER MEMORY BETTER THAN 95% OF THE TIME

- SIMULATION STUDIES SHOWED THAT STORAGE FOR ~128 BLOCKS WAS SUFFICIENT TO LOWER REFERENCES OUTSIDE OF BUFFER STORE TO LESS THAN 5% REGARDLESS OF THE PROGRAM SIZE

- THE ADDRESSING PATTERN OF THE PROGRAM IS THE ONLY SIGNIFICANT CHARACTERISTIC AFFECTING THE EFFICIENCY OF THE BUFFER
HIGH PERFORMANCE HARDWARE

SYSTEM ORGANIZATION - B8500
B8500 CPU - SIMPLIFIED FUNCTIONAL DESCRIPTION

ADVAST

ADDRESS DETERMINATION ADDER

PRTQ
INDEXQ
STOREQ
ASSOCIATIVE MEMORY

FINST

FINST ARITH. & CONTROL

FINQ TEMPQ

STACK EXTENSION

INST. DECODE

(INSTS)

(INSTS)

DATA

DATA

FINST

MGM. ADDR. REG

COMM

MEMORY BUFFER REG.

NO HIT

PRIMARY STORAGE
B8500 INPUT/OUTPUT MODULE BLOCK DIAGRAM

LOCAL MEMORY UNIT
- 1024 Addresses - 104 Bits Per
- 512 Control Channels - 2 Addresses Per

I/O PROCESSING UNIT
- Instructions
- Adders + Comparators
- Field Control
- Relative Addressing Capability (Local Memory + Main Memory)
- Program Flags
- Job Stack Addresses
- Interrupt Stack Addresses

DATA SERVICE UNIT
- Descriptor + Data Register
- Input Data
- Output Data
- Byte Packing + Unpacking
- Service Requests
- Start Lines

COMMUNICATIONS UNIT
- Memory Conflict Resolver
- Input and Output Registers

MEMORY MODULES

CENTRAL PROCESSOR MODULES

PERIPHERAL EQUIPMENT CONTROLLERS
HIGH PERFORMANCE HARDWARE

B8500 - DISC FILE CONTROLLER DETAILED INTERFACE

ELECTRONICS UNIT

QUEUER

CONTROLLER

I/O MODULE
FUNCTIONAL CHARACTERISTICS - B8500 MEMORY MODULE

- 500 NS CYCLE, 200 NS READ ACCESS, 300 NS REGENERATE
- FETCH/STORE 1 OR 4 WORDS

MEMORY BUS (52)

208 BITS (4 X 52)
ELEMENTS OF B8500 CONTRIBUTING TO PERFORMANCE

- MULTIPLE INDEPENDENT MEMORY MODULES
- MULTIPLE PROCESSORS/CHANNELS
- QUEUED ACCESS DISC CONTROLLER
- FUNCTIONAL SEPARATION OF INSTRUCTION PREPARATION, EXECUTION LOGIC, AND COMMUNICATION WITH MEMORY
- INCORPORATION OF ASSOCIATIVE MEMORY FOR
  - INDEX VALUES
  - DESCRIPTORS
  - STORE BUFFER
HIGH PERFORMANCE HARDWARE

SYSTEM DATA INTERCONNECTIONS
A FULL WORD (64 BITS) BIDIRECTIONAL PATH BETWEEN THE PROCESSING ELEMENT AND ITS OWN MEMORY MODULE FOR DATA FETCHING AND STORING.

A PARTIAL WORD (16 BITS), UNIDIRECTIONAL PATH BETWEEN THE PROCESSING ELEMENT AND ITS OWN MEMORY MODULE FOR ALL ARRAY MEMORY ADDRESSING.

A FULL WORD (64 BITS) BIDIRECTIONAL PATH BETWEEN THE PROCESSING ELEMENT AND EACH OF ITS FOUR DESIGNATED ORTHOGONAL NEIGHBORS FOR INTERNETWORK DATA TRANSFERS.

A 2-WORD (256 BITS) UNIDIRECTIONAL PATH BETWEEN EACH MEMORY MODULE AND THE PROCESSING UNIT BUFFER (PUB) FOR TRANSFERS TO IOS AND THE CU.

A 2-WORD (128 BITS) UNIDIRECTIONAL PATH BETWEEN TWO PROCESSING UNITS AND THE PROCESSING UNIT BUFFER FOR INTERQUADRANT ROUTING.

A 1-WORD (64 BITS) UNIDIRECTIONAL PATH BETWEEN THE PROCESSING UNIT BUFFER AND ALL EIGHT PROCESSING UNITS IN THE CABINET (CDB).

A FULL WORD (64 BITS) UNIDIRECTIONAL PATH FROM THE CONTROL UNIT TO EACH OF ITS EIGHT PROCESSING UNIT CABINETS FOR OPERAND BROADCASTING, MEMORY ADDRESSING AND SHIFT COUNT TRANSFERS.

A 200-BIT (APPROXIMATELY) UNIDIRECTIONAL PATH FOR CONTROL UNIT SEQUENCING OF THE PROCESSING ELEMENT QUADRANT.

AN 8-WORD (512 BITS) UNIBIDIRECTIONAL PATH (ONE WORD FROM EACH PUB) FOR DATA TRANSFERS TO THE CONTROL UNIT.

A FULL WORD (72 BITS) BIDIRECTIONAL PATH BETWEEN EACH OF THE FOUR CONTROL UNITS IN THE SYSTEM FOR SYNCHRONIZING AND FOR THE DISTRIBUTION OF COMMON OPERANDS IN THE UNITED ARRAY MODE.

A FULL WORD (64 BITS) BIDIRECTIONAL PATH BETWEEN ADJACENT PROCESSING ELEMENT CABINETS IN ALL FOUR QUADRANTS FOR INTERQUADRANT ROUTING.
A FULL WORD (64 BITS) BIDIRECTIONAL PATH BETWEEN THE FOUR CONTROL UNITS AND THE I/O SUBSYSTEM.

A PART WORD (32 BITS) UNIDIRECTIONAL PATH BETWEEN THE FOUR CONTROL UNITS AND THE I/O CONTROLLER FOR MEMORY ADDRESSING.

A 16-WORD (1024 BITS) BIDIRECTIONAL PATH BETWEEN THE INPUT/OUTPUT SWITCH AND EACH PROCESSING ELEMENT QUADRANT.

A 16-WORD (1024 BITS) BIDIRECTIONAL PATH BETWEEN THE INPUT/OUTPUT SWITCH AND THE I/O SUBSYSTEM.
ILLIAC IV SUBARRAY

CONTROL

CONTROL LINES TO PE's

8.23
DISCUSSION OF ARRAY PROCESSORS

• WHERE DEALING WITH ARRAYS, VERY HIGH PERFORMANCE IS POSSIBLE (UP TO 256 TIMES A VERY HIGH PERFORMANCE SERIAL SYSTEM)

• DATA PLACEMENT CRITICAL IN ILLIAC IV BECAUSE OF LIMITATIONS OF SYSTEM CONNECTIVITY

• INTRODUCES CONCEPT OF PROCESSOR-RELATIVE ADDRESSING.

• CONTROL PROBLEMS COMPOUNDED WHEN INDEXING EXCEEDS DIMENSIONS OF ARRAYS

• ULTIMATE LIMITATION IS HIGHLY PARALLEL ACCESS MEMORY, WITH ILLIAC IV CONNECTIVITY, ONLY 4 PORTS NEEDED FOR EACH MEMORY MODULE

  WITH SAME NUMBER OF PE'S AS A VECTOR CONNECTIVITY EACH MEMORY WOULD REQUIRE 64 PORTS

• EFFICIENCY DEPENDENT ON SOLUTION METHOD ISOMORPHISM WITH STRUCTURE
### PERIPHERAL PROCESSOR MEMORY ALLOCATION

<table>
<thead>
<tr>
<th>PP1-8</th>
<th>PP9</th>
<th>PP0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Temporary Storage</td>
<td></td>
</tr>
<tr>
<td>0075</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Communication Area Addresses</td>
<td></td>
</tr>
<tr>
<td>0100</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Peripheral Resident Program</td>
<td></td>
</tr>
<tr>
<td>0773</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Basic Transient Programs</td>
<td>System Display</td>
</tr>
<tr>
<td></td>
<td></td>
<td>System Monitor</td>
</tr>
<tr>
<td>1773</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Equipment Driver Overlays</td>
<td></td>
</tr>
<tr>
<td>7777</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Resident Central Storage (Typical)

<table>
<thead>
<tr>
<th>TABLE POINTERS</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td><strong>ZEROS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PLD LIMIT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>not used</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>DFB IN</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OUT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LIMIT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FNT LIMIT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>not used</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>EST LIMIT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>not used</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RSL LIMIT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>not used</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CLD LIMIT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>not used</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TRTO</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>last track</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>not used</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TRTI</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>last track</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>not used</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TRT2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>last track</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>not used</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MTR STEP FLAG</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>not used</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Resident Central Storage (Typical) Diagram

- **Control Point Stack Indicators**
- **MTR Temporary Storage**
- **Blank**
- **Blank**
- **PP Starting Times**
- **Central Processor Starting Times**
- **Dates**
- **Times (Hr. Min. Sec)**
- **Simulator P Addr**
- **Simulator XJ Addr**
- **Pseudo-Control Point O Recall IR**
- **PP Idle Time**
- **CP Idle Time**
- **not used**
- **Job Name - MTR**
- **Pseudo-Control Point O**
- **Channel Status Table (CST)**
- **MTR Step Flag**
- **Last Track**
- **Not Used**
- **100G**
- **62G**

### Resident Peripheral Library (RPL)

- **Resident Subroutine Library (RSL)**

### Dayfile Buffer (DFB)

### File Name/File Status Table (FNT/FST)

- **Track Res. Table 2**
- **Track Res. Table 1**
- **Track Res. Table 0**
- **Periph. Lib. Directory**
- **Central Lib. Directory**
- **Equip. Status Table**
- **CP Residents**
- **Control Point Areas**
- **PP Comm. Areas**
- **Pointers and Flags**

---

**9.2**
CONTROL POINT AREAS AND EXCHANGE JUMP AREA

12-BIT BYTE

STATUS

Presence of a "1" bit indicates: the corresponding PP is assigned to this control point

This control point is in recall status

The job at this control point is waiting for the central processor

12-BIT BYTE

ERROR FLAG

-010 Time limit exceeded
010 Arithmetic error
011 PP Abort
100 CP Abort
101 PP Call Error
110 Operator drop
111 Disk Track Limit

Not used

Program Address (P) | AO (Address Registers) |
Reference Address PA | A1 (Increment Register) |
Field Length (FL) | A2 B2 |
Exit Mode (EM) | A3 B3 |
A4 B4 |
A5 B5 |
A6 B6 |
A7 B7 |
X0 (Operand Registers)
X1
X2
X3
X4
X5
X6
X7

STATUS | ERROR FLAG | program move flag | RA (Hundred) | FL (Hundred) |
JOB NAME (DISPLAY CODE) | next cont. stat. |
PRIORITY | MSG.COUNT | TRACK COUNT | TIME LIMIT | pp assign equip |
CP TIME (SECS) | (MSECS) |
PP TIME (SECS) | (MSECS) |
PP RECALL REG.
SENSE SWITCHES, LIGHTS
EQUIPMENT ASSIGNED
LAST DAYFILE MESSAGE |
(OR CONSOLE MESSAGE) |

CONTROL STATEMENT BUFFER |
(PACKED DISPLAY CODE) |

RECALL REG.

EXCHANGE PACKAGE

POINTER TO NEXT STATEMENT IN BUFFER

CURRENT RUNNING TIMES

HOLDS PP INPUT REGISTERS DURING PP RECALL

EQUIPMENT ASSIGNMENT

59

4 17 20 33 34 47 50 63 64 77
48 47 36 36 24 23 12 11

9.3
The Transient Program may also initiate MTR Requests via PP Resident
MB Message Buffer
IR Input Register
OR Output Register
CP Control Point
ARG Argument

* The Transient Program may also initiate MTR Requests via PP Resident
MB Message Buffer
IR Input Register
OR Output Register
CP Control Point
ARG Argument

9.4
DAYFILE DISPLAY

This column represents the time each control statement was requested for execution. (A total of 32 lines may be contained on the dayfile.)

New dayfile information appears at the bottom of the screen automatically; old dayfile information is deleted at the top of the column as new times are entered into the dayfile.

This column represents the control statements introduced via card input and contains the system's history. A summary of the day's total run may be printed out upon request.

NOTE: Dayfile display data will appear on the printout at the end of each job automatically.
A simulator in use indicates an S in place of P.

- **FIELD NAME**
- **FIELD LENGTH**
- **PROGRAM ADDRESS**
- **CHANNEL ASSIGNMENT**
- **EQUIPMENT**
- **CHANNELS**
- **STATUS OF CHANNELS**
  - D - Disconnected
  - E - Empty
  - F - Full
- **PROGRAM STATUS**
  - Blank
  - W
  - X
  - A-G
- **TIME LIMIT** (OCTAL SECONDS)
- **ACTUAL CENTRAL PROCESSOR TIME** (OCTAL SECONDS)
- **PERIPHERAL PROCESSOR** (PP5 ASSIGNED TO THIS CONTROL POINT)
- **EQUIPMENT ASSIGNMENTS**
- **LAST PROGRAM MESSAGE** (NAME OF PP PROGRAM BEING RUN)
<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>PROGRAMMING</th>
<th>CHECKOUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>010000</td>
<td>26730</td>
<td>62330</td>
</tr>
<tr>
<td>010001</td>
<td>30645</td>
<td>26010</td>
</tr>
<tr>
<td>010002</td>
<td>07125</td>
<td>06232</td>
</tr>
<tr>
<td>010003</td>
<td>32610</td>
<td>70401</td>
</tr>
<tr>
<td>010004</td>
<td>20002</td>
<td>06602</td>
</tr>
<tr>
<td>010005</td>
<td>02000</td>
<td>76101</td>
</tr>
<tr>
<td>010006</td>
<td>00000</td>
<td>00000</td>
</tr>
<tr>
<td>010007</td>
<td>35022</td>
<td>40000</td>
</tr>
<tr>
<td>010010</td>
<td>01000</td>
<td>00002</td>
</tr>
<tr>
<td>010011</td>
<td>12020</td>
<td>51430</td>
</tr>
<tr>
<td>010012</td>
<td>02002</td>
<td>56130</td>
</tr>
<tr>
<td>010013</td>
<td>24010</td>
<td>32100</td>
</tr>
<tr>
<td>010014</td>
<td>02002</td>
<td>56102</td>
</tr>
<tr>
<td>010015</td>
<td>10660</td>
<td>56730</td>
</tr>
<tr>
<td>010016</td>
<td>30615</td>
<td>46334</td>
</tr>
<tr>
<td>010017</td>
<td>30625</td>
<td>41330</td>
</tr>
<tr>
<td>010020</td>
<td>10063</td>
<td>15510</td>
</tr>
<tr>
<td>010021</td>
<td>62103</td>
<td>06434</td>
</tr>
<tr>
<td>010022</td>
<td>30531</td>
<td>00631</td>
</tr>
<tr>
<td>010023</td>
<td>16036</td>
<td>21001</td>
</tr>
<tr>
<td>010024</td>
<td>00000</td>
<td>00000</td>
</tr>
<tr>
<td>010025</td>
<td>00003</td>
<td>03102</td>
</tr>
<tr>
<td>010026</td>
<td>76000</td>
<td>20023</td>
</tr>
<tr>
<td>010027</td>
<td>30725</td>
<td>40430</td>
</tr>
<tr>
<td>010030</td>
<td>74002</td>
<td>00050</td>
</tr>
<tr>
<td>010031</td>
<td>04113</td>
<td>40137</td>
</tr>
<tr>
<td>010032</td>
<td>04020</td>
<td>67303</td>
</tr>
<tr>
<td>010033</td>
<td>02002</td>
<td>30112</td>
</tr>
<tr>
<td>010034</td>
<td>23011</td>
<td>22004</td>
</tr>
<tr>
<td>010035</td>
<td>07513</td>
<td>62210</td>
</tr>
<tr>
<td>010036</td>
<td>21000</td>
<td>00036</td>
</tr>
<tr>
<td>010037</td>
<td>30321</td>
<td>63076</td>
</tr>
</tbody>
</table>

**C, D, E DISPLAYS**

- [ ]
- [ ]
- [ ]
- [ ]
- [ ]

**F, G DISPLAYS**

- [ ]
- [ ]
- [ ]
- [ ]
- [ ]

**H DISPLAY**

- [ ]
- [ ]
- [ ]
- [ ]
- [ ]

**Display Area**

For Operator Typing
(Always on left screen)

**Contents of Address**

- High Performance Software
- Storage Displays

**Four Groups of Five-Octal Digits**

- (Or five groups of four-octal digits)

- Jobs stacked for output

(If no jobs are waiting for output none will appear)
HIGH PERFORMANCE SOFTWARE

JOB FLOW

PROGRAMMER INPUT TAPES

I. READ JOHNO05

PROGRAMMER OUTPUT TAPES

H. DISPLAY

A DISPLAY

INPUT OUTPUT...JOHNO05

INPUT OUTPUT JOHNO05

CONTROL POINT 4

A DISPLAY

4. JOHNO05 REQUEST TAPE

DISK

LINE PRINTER

CARD PUNCH

2. OUTPUT

1. PRINT 2 PRINT 3. PUNCH JOHNO05 IDLE JOHNO05
TYPICAL DECK SEQUENCE

Card arrangement to begin a job, separate records, and terminate.
Card arrangement for a FORTRAN Load and Run job:

Tape references:

TAPE1 - assumed input tape which operator loads on a particular unit

TAPE5 - output tape drawn from tape pool

TAPE6 - scratch file on disk
SYSTEMS COMPONENTS

DISK
EXECUTIVE
AND MONITOR PP

EXECUTIVE
CONTROLS SYSTEM
EVALUATES PRIORITIES
SCHEDULES JOBS
MAINTAINS I/O REQUESTS
PROVIDES CONSOLE DISPLAY
MONITOR MONITORS I/O REQUESTS
CHECKS I/O STATUS

COMPUTATIONAL LOAD

CENTRAL MEMORY

CENTRAL PROCESSOR

JOBS

JOB 1
JOB 2
JOB 3

POOL PP RESIDENTS
PICK UP I/O REQUESTS
LOAD JOBS
REPORT COMPLETION

PROGRAMMER SCRATCH AREA
WORK AREA

JOB STACK
JOBS TO BE ASSEMBLED
JOBS TO BE COMPILED
JOBS TO BE EXECUTED

OUTPUT BUFFERS
PRINT DATA
PUNCH DATA

SYSTEM ROUTINES
ALL ROUTINES

SYSTEM DISK

wałink & SAVES

LDI LINE PRINTER
CONSOLE DISPLAY
CARD READER
AND PUNCH

HIGH PERFORMANCE SOFTWARE
JOB CONTROL

CONTROL CARDS
(*REQUIRED CONTROL CARDS)

JOB IDENTIFICATION
- JOB NAME AND ACCOUNT NUMBER
- PRIORITY
- CENTRAL PROCESSOR RUNNING TIME LIMIT

EQUIPMENT
- SCRATCH TAPE
- INPUT TAPE
- OUTPUT TAPE
- PRINTER
- DISK
- CARD READER
- CARD PUNCH
- PERIPHERAL PROCESSOR

VARIATIONS
- VARIABLE vs FIXED REQUIREMENTS
- EQUIPMENT EXCHANGE
- SPECIFIC ASSIGNMENT

MEMORY ESTIMATE
- CENTRAL MEMORY
  - FIXED
  - VARIABLE
- DISK MEMORY
  - FIXED
  - VARIABLE

DEBUGGING
- MEMORY DUMP
- MEMORY MAP
- CONSOLE DEBUGGING
- ERROR HALT CONDITIONS

OTHER
- IGNORE EXPONENT OVERFLOW
- IGNORE INDEFINITE RESULT
- IGNORE EXPONENT OVERFLOW AND INDEFINITE RESULT
- COMPILE PROGRAM
  + FINIS

CARD DECK LAYOUT

[Diagram of card deck layout]

- END-OF-JOB CARD
- DATA CARDS
- PROGRAM CARDS
- OPTIONAL CONTROL CARDS
- REQUIRED CONTROL CARD

9.13
HIGH PERFORMANCE SOFTWARE

JOB DECK EXAMPLES

CASE A: JOB COMPILATION

CASE B: JOB COMPILATION AND EXECUTION

CASE C: JOB EXECUTION (NO OPTIONAL CARDS)

CASE D: JOB EXECUTION (OPTIONAL CONTROL CARDS)
USE OF MEMORY

COMBINATIONS OF ABOVE ARE POSSIBLE

1. JOB TO SYSTEM
2. SYSTEM SCHEDULES COMPILATION
3. JOB COMPILED — COMPILED JOB BUCK TO JOB STAGE
4. SYSTEM SCHEDULES EXECUTION
5. EXE JOB (INCLUDING IF PROGRAM) TO CP
6. WHEN NEEDED IS NEXT SYSTEM EXHIBIT + JUMP TO CP
7. IF PROGRAMMER MANOE INSTRUCTED CP CP PROGRAM
8. EXE EXECUTED BY CP SPLIT CP PROGRAM
9. CP PROGRAMS SPLIT IN CP PROGRAM
10. WHEN NEEDED IS NEXT SYSTEM EXHIBIT + JUMP TO CP
11. CP PROGRAMS SPLIT IN CP PROGRAM
12. WHEN NEEDED IS NEXT SYSTEM EXHIBIT + JUMP TO CP
13. CP PROGRAMS SPLIT IN CP PROGRAM
14. WHEN NEEDED IS NEXT SYSTEM EXHIBIT + JUMP TO CP
15. CP PROGRAMS SPLIT IN CP PROGRAM

USE EXECUTION AND BALANCE OF POOL PnP FUNCTIONAL SYSTEM FUNCTIONS.

COMBINATIONS OF ABOVE

ARE POSSIBLE

9,15
HIGH PERFORMANCE SOFTWARE

PROCESSING STEPS

1. BATCH LOADER LOADS JOB INTO JOB STACK ON DISK FROM CARDS OR TAPE MAKES ENTRY IN JOB TABLE FOR EACH JOB LOADED.

2. EXECUTIVE EXAMINES JOB TABLE FOR JOBS TO BE LOADED INTO CM INSTRUCTS JOB LOADER TO LOAD JOB WITH HIGHEST PRIORITY (IF IT MEETS LOADING REQUIREMENTS) MAKES EQUIPMENT ASSIGNMENTS IN EQUIPMENT TABLE REQUESTS OPERATOR TO PREPARE EQUIPMENT.

3. EXECUTIVE EXCHANGE JUMPS TO JOB TO BE EXECUTED.
PROCESSING STEPS (CONT.)

4. EXECUTIVE instructs job loader to load other jobs into central memory until it is full. Multiprocesses jobs in central memory.

5. EXECUTIVE directs output data for printer and punch to output buffer on disk. Directs output data for tape to pool PP which writes tape.

6. EXECUTIVE schedules new job for CM when job terminates. Instructs job loader to load new job from job stack on disk into CM.
REMOTE HOOKUPS

- CENTRAL PROCESSOR
- CENTRAL MEMORY
- OTHER PROGRAMS
- SYSTEM DISK
- SYSTEM REQUIREMENTS

CASE 1: CP PROGRAM
CASE 2: CP PROGRAM
CASE 3: DUMMY CP PROGRAM

SYSTEM AND POOL PP FOR NORMAL SYSTEM FUNCTIONS
NORMAL IN
NORMAL OUT

CASE 1 -- AUXILIARY
SPECIAL PP PROGRAM PERFORMING AUXILIARY FUNCTION FOR CP PROGRAM
NO I/O INVOLVED
EXAMPLE: EDITING

CASE 2 -- REAL TIME
SPECIAL PP PROGRAM PERFORMING CONTROL AND TRANSMISSION OF REAL TIME INFORMATION FOR CP PROGRAM.

CASE 3 -- REMOTE STATION
SPECIAL PP PROGRAM FUNNELING NORMAL JOBS IN AND OUT OF SYSTEM.

SYSTEM AND POOL PP
FOR NORMAL SYSTEM FUNCTIONS

JOB STACK
V/O AREA
PROGRAMMER "SCRATCH" AREA

NORMAL JOBS FROM REMOTE STATIONS
REAL TIME DEVICE
MULTIPLEXER

REAL TIME DEVICES

OUTPUT TO REMOTE STATIONS

NORMAL JOBS FROM REMOTE STATIONS
SCOPe FEATURES

• CHIPPEWA SUPERVISOR

• DATA MANAGEMENT

• ADVANCED LOADER

• REMOTE PACKAGE
# Storage Assignment During Segmentation

<table>
<thead>
<tr>
<th>Loading Order</th>
<th>Segment Level</th>
<th>Contents of User's Job Area in Memory after Loading of Segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>SEG 0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>SEG 0, SEG 3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>SEG 0, SEG 3, SEG 4</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>SEG 0, SEG 3, SEG 4, SEG 9</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>SEG 0, SEG 2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>SEG 0, SEG 1</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>SEG 0, SEG 1, SEG 5</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>SEG 0, SEG 1, SEG 5, SEG 8</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>SEG 0, SEG 1, SEG 5, SEG 7</td>
</tr>
</tbody>
</table>

Unused Storage Area
# Storage Assignments for Overlays

<table>
<thead>
<tr>
<th>Loading Order</th>
<th>Primary Level Number</th>
<th>Secondary Level Number</th>
<th>Contents of User's Job Area in Memory after Loading of Overlay</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>(0,0)</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>(0,0) (1,1)</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>(0,0) (1,0) (1,1)</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>(0,0) (1,0) (1,2)</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>(0,0) (1,0) (1,1)</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3</td>
<td>(0,0) (1,0) (1,3)</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>2</td>
<td>(0,0) (1,0) (1,2)</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>0</td>
<td>(0,0) (2,0)</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>1</td>
<td>(0,0) (2,0) (2,1)</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>2</td>
<td>(0,0) (2,0) (2,2)</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>0</td>
<td>(0,0) (3,0)</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>0</td>
<td>(0,0) (4,0)</td>
</tr>
</tbody>
</table>
FILE ENVIRONMENT TABLE (FET)

<table>
<thead>
<tr>
<th>Bits 59</th>
<th>47</th>
<th>44</th>
<th>35</th>
<th>32</th>
<th>29</th>
<th>23</th>
<th>17</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>logical file name (lfn)</td>
<td>code and status</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>device type</td>
<td>r</td>
<td>m</td>
<td>u</td>
<td>d</td>
<td>disposition code</td>
<td>FIRST</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IN</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FNT pointer</td>
<td>record block size</td>
<td>physical record unit size</td>
<td>LIMIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>working storage fwa</td>
<td>working storage lwa+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>record number</td>
<td>index length</td>
<td>index address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EOI address</td>
<td>error address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Label file name (first 10 chars)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Label file name (last 10 chars)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>edition number</td>
<td>retention cycle</td>
<td>creation date</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>position number</td>
<td>multi-file name</td>
<td>reel number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
BUFFERING DURING A READ

INITIAL STATE

AFTER READ

AFTER PROCESS

AFTER READ
RESPOND COMMANDS TO SCOPE

- COMPILE
- ASSEMBLE
- EXECUTE
- COPY
- SUBMIT
SYSTEM ACTION REQUESTS

- MEMORY
- CKPT
- RECALL
- MESSAGE
- ENDRUN
- ABORT
- LOADER
- TIME/DATE
FILE ACTION REQUESTS

- REQUEST
- OPEN
- CLOSE
- EVICT
- READ
- WRITE
- SKIP
- BKSP
- REWIND
- UNLOAD
A RESPOND DIALOGUE

LOGIN JRV, 2359
CONTINUE

FORMAT FTN 80 TAB 2,7
CONTINUE

INPUT FTN

0010 \# PROGRAM EOQ (INPUT= TAPE1,OUTPUT=TAPE2)
0020 \# LU1=TAPE1
0030 \# LU2=TAPE2
0040 \# READ(LU1,10)USE,POC,UC
0050 \# FORMAT (3F8.2)
0060 \# IF (USE.EQ.7777)40,30
0070 \# CONTINUE
0080 . .
.
.

III0 \# CALL REPORT(QTY,POC,UCOST, TCOAST)
.
.

IIJ0 \# RETURN
IIK0 \# END
III0 \# EOF
CONTINUE

FILE EOQ, 10 TO III0
CONTINUE

COMPILE EOQ
Job name from SCOPE
Notification of job completion

LIST FILES
PRIVATE FILES
TAPE1 150 DIS 80 1 1/1/67
EOQ 130 DIS 80 1 3/1/67
EOQ L 230 DIS VL 1 2/12/67
EOQ B 52 BIN 20 1 2/12/67

9.28
A RESPOND DIALOGUE

EXECUTE EOQ B, INPUT=TAPE1,
OUTPUT=TAPE2 Δ
Job name from SCOPE
Notification of job completion

OPEN TAPE1 Δ
CONTINUE

DISPLAY RECORD 1 TO 5 Δ

ORDER PO UNIT TOTAL
QTY  COST  COST  COST
942  100.00 120.00 113140.00
330  8.00  33.50  110663.00
481  1.20  9.80  4715.00

DISPLAY RECORD 5 TO 10 Δ

481  1.20  9.80  4715.00
366  1.80  5.50  2014.80

TOTAL 358320.15

DISPLAY RECORD 1, 2, 5 TO 8 Δ

ORDER PO UNIT TOTAL
QTY  COST  COST  COST
481  1.20  9.80  4715.00
366  1.80  5.50  2014.80

TOTAL 358320.15

OPEN FILE EOQ L Δ
CONTINUE

DELETE EOQ L Δ
CONTINUE

COPY TAPE2 TO PRINTER Δ
Job name from SCOPE
Notification of job completion

LIST FILES Δ
PRIVATE FILES
TAPE1 150 DIS  80 2 1/1/67
EOQ  130 DIS  80 1 3/1/67
EOQ B  52 BIN  20 1 2/12/67
TAPE2  8 DIS  80 3 2/12/67

LOGOUT Δ
TIME 00.35.05
A RESPOND DIALOGUE

LOGIN GFC, 2106 Δ
. CONTINUE

INPUT Δ
0010   ASPER MUX
0020   TERM EQU 12 Δ
0030   CHAN EQU 13B Δ
0040   CONN EQU 5001B Δ
0050

iil0   IOP FNC CHAN, CONN Δ.

iij0
iij0 + EOF Δ
CONTINUE

FILE MUXIO Δ
CONTINUE

ASSEMBLE MUXIO Δ
Job name from SCOPE
Notification of job completion

LIST FILES Δ
PRIVATE FILES
MUXIO 122 DIS 80 1 2/10/67
MUXIO L 352 DIS VL 1 2/10/67
MUXIO B 37 BIN 20 1 2/10/67

LOGOUT Δ
TIME 00.13.20
TOPICS COVERED THIS SESSION

- THE INFLUENCE OF PROGRAMMING LANGUAGE ON MACHINE DESIGN - PARTICULARLY THE EFFECT OF ALGOL 60
- SEVERAL MACHINE DESIGNS REFLECTING THIS INFLUENCE
  
  B 5000
  
  KDF 9
  
  B 55/65/7500
NEW NOTIONS IN ALGOL 60

- ORIGINS IN ALGOL '58
  PRODUCED BALGOL
  MAD
  NELLIAC
  JOVIAL

- BLOCK STRUCTURE
  (STATIC LEVELS)

- RECURSION IN PROCEDURES
  (DYNAMIC LEVELS)

- MIXED MODE ARITHMETIC
BLOCK STRUCTURE AND STORAGE ALLOCATION

BLOCK A

A

B

E

G

F

D

I

J

C

H

a.

BLOCK STRUCTURE AS WRITTEN

b.

TREE FORM FOR CODE

PROGRAM STORAGE = MAX (ACH, ADIJ, ABF, ABEG)
TO BE RESERVED

10.3
SAMPLE ALGOL PROGRAM WITH BLOCK STRUCTURE AND SUBROUTINES

A: BEGIN REAL SCR, THETA; REAL ARRAY VAL, (1:29);
INTEGER ARRAY M (1:50, 1:15), PLT (1:50, 1:15), V (1:29);
INTEGER i, j, k, n, p, q, score, length, wd, rnk;

PROCEDURE B (m, n, PLT);
VALUE m, n, e;
BEGIN INTEGER i, s, n;
FOR i := 1 STEP 1 UNTIL N DO
BEGIN FOR j := 1 STEP 1 UNTIL 2 DO
BEGIN k := K + 1;
PLT [k] := PLT [i, i]. 1000000 END;
k := [k] END END END B:

PROCEDURE C (length, score, q, plt);
VALUE length, q;
BEGIN INTEGER t, u;
t := length . q;
B(t, u, length, PLT);
score := PLT/u end C;

IF (PLT [i] ≠ 0) AND (PLT [i] ≠ wd) then
go to D else if PLT [i] > SCR then
go to E else

t := rnk [k];
B(t, e, q, m);
C(i, k, p, m);

D: BEGIN REAL k;
[l] := k;
val[j] := k;
i := j + 1;
k := 1 end D;

E: BEGIN REAL k;

PROCEDURE F (j, k);
VALUE j;
k := j 5 end F;
q := n . p;
F(q, wd);
t := q/lgth. end E;

END A AND PROGRAM;

10.4
BLOCK STRUCTURE OF SAMPLE PROGRAM

A
  *B
  *C
  D
  E
  *F
STRUCTURE OF PROGRAM A ON TAPE, REARRANGEMENT SUPPLIED BY COMPILER

<table>
<thead>
<tr>
<th>SWITCH TABLE FOR A</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECL. of A</td>
</tr>
<tr>
<td>DECLARATIONS AND BODY OF R</td>
</tr>
<tr>
<td>DECLARATIONS AND BODY OF C</td>
</tr>
<tr>
<td>BODY OF A</td>
</tr>
<tr>
<td>DECL. OF D</td>
</tr>
<tr>
<td>BODY OF D</td>
</tr>
<tr>
<td>DECLARATIONS OF E</td>
</tr>
<tr>
<td>DECLARATIONS AND BODY OF F</td>
</tr>
<tr>
<td>BODY OF E</td>
</tr>
</tbody>
</table>

10.6
MEMORY ALLOCATION AFTER INITIAL LOADING

- L
- SUBROUTINE B
- SUBROUTINE C
- C
- CODE BODY OF A
- P
- ARRAY STORAGE FOR A
- K
- ARITHMETIC STACK FOR A
- SB
- DECLARATIONS FOR A
  SWITCH TABLE FOR A (B, C)
MEMORY AFTER CALL ON B

- **L**
  - SUBROUTINE B
  - SUBROUTINE C
- **C**
  - CODE BODY OF A
  - ARRAY STORAGE OF A
  - RETURN POINT IN A, SB FOR A
- **P**
  - ARRAY STORAGE OF B

- **K**
  - ARITHMETIC STACK FOR B
  - DECLARATIONS FOR B
- **SB**
  - ARITHMETIC STACK FOR A
  - DECLARATIONS FOR A
  - SWITCH TABLE FOR A

10.8
MEMORY AFTER CALL ON C

- L
- SUBROUTINE B
- C
- SUBROUTINE C
- CODE BODY OF A
- ARRAY STORAGE FOR A
- P
- RETURN POINT IN A, SB FOR A
- RETURN POINT IN C, SB FOR C
- K
- ARITHMETIC STACK FOR B
- DECLARATIONS FOR B
- ARITHMETIC STACK FOR C
- DECLARATIONS FOR C
- ARITHMETIC STACK FOR A
- DECLARATIONS FOR A
- SWITCH TABLE FOR A
ENTRY INTO NEW BLOCK D

CONTROL WORD, RETURN POINT IN A/, SB IN A/P

SUBROUTINE B

SUBROUTINE C

CODE BODY OF A

ARRAY STORAGE FOR A

CODE BODY OF D

ARITHMETIC STACK FOR D

DECLARATIONS FOR D

SWITCH TABLE FOR D

ARITHMETIC STACK FOR A

DECLARATIONS FOR A

SWITCH TABLE FOR A
MEMORY AFTER CALL ON F WITHIN E

L

CONTROL WORD, RETURN POINT IN A, SB FOR A/P

SUBROUTINE B

SUBROUTINE C

C

CODE BODY OF A

ARRAY STORAGE FOR A

SUBROUTINE F

CODE BODY FOR E

P

CONTROL WORD, RETURN POINT IN E, SB FOR E

K

ARITHMETIC STACK FOR F

DECLARATIONS FOR F

ARITHMETIC STACK FOR E

DECLARATIONS FOR E

SB

SWITCH TABLE FOR E

ARITHMETIC STACK FOR A

DECLARATIONS FOR A

SWITCH TABLE FOR A
RECURSION IN AN UNRELATED PROGRAM

L
--- SUBROUTINE R
--- SUBROUTINE S
--- CODE BODY Q

C
--- RET. POINT IN Q, SB FOR Q
--- RET. POINT IN R, SB FOR R
--- RET. POINT IN S', SB FOR S'
--- RET. POINT IN R₂, SB FOR R₂
--- RET. POINT IN S₂, SB FOR S₂

P
--- A.S. FOR R₃
--- DECL. FOR R₃
--- A.S. FOR S₂
--- DECL. FOR S₂
--- A.S. FOR R₂
--- DECL. FOR R₂
--- A.S. FOR S₁
--- DECL. FOR S₁
--- A.S. FOR R₁
--- DECL. FOR R₁
--- ARITHMETIC STACK FOR Q

K

SB
--- DECLARATIONS FOR Q
--- SWITCH TABLE FOR Q (R, S)

STORAGE CREATED AT RUN TIME
POLISH NOTATION AND ARITHMETIC EXPRESSIONS

\[(A + (B - C)) \times (D/E + F)\]

1 3

2 4

EVALUATION ORDER

\[\begin{array}{c}
A \\
B \\
C \\
D \\
E \\
F \\
+ \\
- \\
\times \\
\div \\
\end{array}\]

TREE REPRESENTATION OF EXPRESSION

A, B, C, -, +, F, D, E, \/, +, X
B, C, -, A, +, D, E, \/, F, +, X

EQUIVALENT SUFFIX POLISH FORMS FOR EXPRESSION

X, +, -, B, C, A, \/, D, E, F

POLISH PREFIX FORM
STACK MACHINES AND OTHER ADVANCED SYSTEMS CONCEPTS

FLOWCHART FOR CONVERTING EXPRESSIONS TO SUFFIX POLISH FORM BASED ON OPERATOR HIERARCHY

1. START
2. i → OPSTAC
3. GET NEXT ITEM FROM SOURCE LANGUAGE
4. IS IT AN OPERAND?
   - No: IS OPSTAC A (?)
   - Yes: ITEM → ANDSTAC
5. IS ITEM a ;?
   - No: IS OPSTAC A (?)
   - Yes: ITEM → OPSTAC
6. IS OPSTAC a (?)
   - No: ITEM → OPSTAC
   - Yes: ITEM → OPSTAC
7. IS ITEM a ;?
   - No: ITEM → OPSTAC
   - Yes: ITEM → OPSTAC
8. ANDSTAC → POLISH STRING
   - (PR) → ANDSTAC
9. ANDSTAC → POLISH STRING
10. OPSTAC : ITEM
11. ITEM → OPSTAC
12. OPSTAC : ITEM
13. ITEM → OPSTAC

Symbols:
- 'GOES INTO'
- OPSTAC: AN OPERATOR STACK
- ANDSTAC: OPERAND ADDRESS STACK
- ≠: EQUAL PRECEDENCE
- >: GREATER PRECEDENCE
- <PR>: SYMBOL FOR PARTIAL RESULT
DATA FETCH AS A SEPARATE OPERATOR

| CLA  | B       | FETCH B   |
| SUB  | C       | FETCH C   |
| ADD  | A       | SUB       |
| STO  | TEMP    | FETCH A   |
| CLA  | D       | ADD       |
| DIV  | E       | FETCH D   |
| ADD  | F       | FETCH E   |
| MUL  | TEMP    | DIV       |
|      |         | ADD       |
|      |         | MUL       |

SINGLE ADDRESS INST
PROGRAM FOR
EXPRESSION

'STACK' PROGRAM
FOR
EXPRESSION
FUNCTIONS IN STACK MACHINE

\[ \text{SIN} (X) \]
\[ \text{MAX} (a, b, c, d, e \ldots) \]

FUNCTIONS

\[ X, \text{SIN}, <\text{SRE}> \]
\[ a, b, c, d, e \ldots, \text{MAX}, <\text{SRE}> \]

POLISH FORM
STACK AS A COMMUNICATIONS MEDIUM

SUBRA (A, B, C)

<table>
<thead>
<tr>
<th>A</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

WORK SPACE
FOR SUBRA
STACK MACHINES AND OTHER ADVANCED SYSTEMS CONCEPTS

SIMPLIFIED FUNCTIONAL DIAGRAM OF B5000 ORGANIZATION

ARITHMETIC AND CONTROL

TOP OF STACK
A

2ND POSITION OF STACK
B

STACK EXTENSION IN PRIMARY STORAGE

F
S
# INSTRUCTIONS IN B5000

<table>
<thead>
<tr>
<th>BIT 10</th>
<th>BIT 11</th>
<th>SYLLABLE TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>LITERAL CALL</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>OPERATOR</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>OPERAND CALL</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>DESCRIPTOR CALL</td>
</tr>
</tbody>
</table>
## EFFECT OF OPERAND CALL SYLLABLE IN B5000

<table>
<thead>
<tr>
<th>Type of Word Accessed</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operand</td>
<td>Place in top of stack</td>
</tr>
<tr>
<td>Control Word</td>
<td>Place in top of stack, treat as an operand</td>
</tr>
<tr>
<td>Data Descriptor</td>
<td>Word addressed by descriptor placed in TOS, treated as an operand</td>
</tr>
<tr>
<td>Program Descriptor</td>
<td>Place a return control word in TOS, branch to subroutine</td>
</tr>
</tbody>
</table>
### Effect of Descriptor Call Syllable in B5000

<table>
<thead>
<tr>
<th>Type of Word Accessed</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operand</td>
<td>Generate a data descriptor with absolute address of operand and place in TOS</td>
</tr>
<tr>
<td>Data Descriptor</td>
<td>Place data descriptor in TOS</td>
</tr>
<tr>
<td>Program Descriptor</td>
<td>Place a return control word in TOS, branch to subroutine</td>
</tr>
</tbody>
</table>
THE KDF-9 COMPUTER SYSTEM

PRIMARY STORAGE

N1

N2

N3

ARITHMETIC AND CONTROL

16 WORD NESTING STORE

16 WORD SJNS

16 WORD Q STORE
VARIABLE LENGTH INSTRUCTIONS IN KDF-9

- 0 7: ARITHMETIC OPERATORS
- SYL SYL: SHIFT INSTRUCTIONS, I/O OPERATORS
- SYL SYL SYL: MEMORY FETCH, STORE, JUMPS
SUMMARY OF STACK MACHINE DESIGN PRINCIPLES

- STACK CONCEPT PROVIDES 'AUTOMATIC AND ANONYMOUS' TEMPORARY STORAGE
- STACK PROVIDES DYNAMIC STORAGE ALLOCATION FOR NESTED AND RECURSIVE SUBROUTINES
- POLISH NOTATION SUGGESTS SYLLABIC INSTRUCTION FORMATS
- SEPARATE FETCH AND STORE OPERATORS PERMITS HARDWARE DETECTION AND INTERPRETATION OF CONTROL WORDS AND DESCRIPTORS
- STACK MACHINES SIMPLIFY COMPILING BECAUSE INTERNAL STRUCTURE MATCHES A 'NATURAL' INTERMEDIATE LANGUAGE, AND ELIMINATES NEED TO KEEP TRACK OF TEMPORARY STORAGE
STACK MACHINE EXECUTIVES AND PRECISION CONSIDERATIONS

THIS SESSION WILL COVER

- OPERATING SYSTEMS OVERHEAD
- OPERATING SYSTEMS DESIGN FOR 'STACK' MACHINES
- PRECISION IN COMPUTERS
DISTRIBUTION OF FUNCTIONS IN OPERATING SYSTEMS

FUNCTIONS REQUIRED BY USER IN EXECUTION OF THIS PROGRAM(S)

- I/O
- SUPERVISORY SERVICES (OBTAINING OVERLAYS, EXECUTION OF COMMON SUBROUTINES)

FUNCTIONS TRANSPARENT TO USER

- MEMORY ALLOCATION/DEALLOCATION
- SCHEDULING/DISPATCHING
- INTERRUPT SERVICING
- SWAPPING (IF PRESENT)
SOURCES OF OVERHEAD

- SPACE REQUIRED BY OPERATING SYSTEM (RESIDENT AND NON-RESIDENT)
- TIME REQUIRED TO RE-DIRECT CPU FOR INTERRUPT PROCESSING
- SWAPPING FOR CONVENIENCE OF OPERATING SYSTEM
METHODS FOR REDUCING OVERHEAD

- 'WIRED-IN' OPERATING SYSTEMS MICROPROGRAMMING
  DEDICATED STORAGE FOR SYSTEM TABLES
- MULTIPLE CONTROL STATES WITH SEPARATE STATE WORDS
  RCA SPECTRA 70 SERIES
  SDS SIGMA 7
- ASSOCIATIVE STORAGE FOR SCRATCHPAD REGISTERS
- INDEPENDENT CHANNELS
- HIGH SPEED BULK STORAGE
  LCS
  QUEUE DRIVEN ROTATING STORAGE
OPERATING SYSTEM CONCERNS FOR STACK MACHINES

- DYNAMIC STORAGE ALLOCATION FOR BLOCK STRUCTURES
- DYNAMIC STORAGE ALLOCATION FOR STACK EXTENSION INTO PRIMARY STORAGE
  - RECURSIVE SUBROUTINES
  - DYNAMIC ARRAYS
  - ARITHMETIC STACK
B5500 PROGRAM STRUCTURE AND PRT

A
D₁
D₂
D₃

B
D₁
D₂
D₃

D₁
D₂
D₃

F
D₁
D₂
D₃

PRT
A
D₁(A)
D₂(A)
D₃(A)
B
D₁(B)
D₂(B)
E
D₁(E)
D₂(E)
D₃(E)
F
D₁(F)
D₂(F)
D₃(F)
ADDRESSING HIGHER LEVEL BLOCK DATA

\[(rR) + \text{PRT} \cdot \text{REL ADDRESS OF A} + \text{INDEX OF D_3} \]
STACK MACHINE EXECUTIVES AND PRECISION CONSIDERATIONS

B5500 STACK STRUCTURE FOR SUBROUTINES

rF

rS

TOS-2

rF

rS

TOS-2

(rF)

P1

P2

P3

(rL)(rF)(rC)

LV1

LV2

MARK STACK CONTROL WORD

RETURN CONTROL WORD
DF MCP Classification of Primary Storage

- Non-Overlayable Storage
  - Resident MCP
  - System Tables
  - Program PRT and Stack Areas

- Overlayable Storage
  - Program Segments
  - Data Areas (Arrays)

- Available Storage
DFMCPRGANIZATION OF PRIMARY STORAGE

PRIMARY STORAGE
DF MCP PROCEDURES

- STATUS
- CONTROL CARD
- SELECTION
- RUN
- INITIATE
- PRESENCE BIT
DF MCP CONTROL PROCEDURES

- SLEEP
- NOTHINGTODO
- GETSPACE
- OLAY
- FORGETSPACE
- ESPBIT
B5500 PARALLEL PROCESSING AND CHECKPOINT FACILITIES

- PARALLEL PROCESSING AND PRIORITY INTERRUPTS
- BREAKOUT, RESTART, EMERGENCY INTERRUPT
CHARACTERISTICS OF B5500 OP. SYSTEM.

- Multiprogramming designed in at the start
- Provides multiprocessing control
- Provides dynamic storage allocation for program segments data (arrays)
- Stack mechanism handles recursive subroutines arithmetic stack
- Supports on-line use
  the interp system
  datacomm system
STACK MACHINE EXECUTIVES AND PRECISION CONSIDERATIONS

PRECISION COMPARISONS

<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>INTEGERS</th>
<th>SINGLE FLOATING</th>
<th>DOUBLE FLOATING</th>
<th>S/H</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CHAR.</td>
<td>MANT.</td>
<td>CHAR.</td>
</tr>
<tr>
<td>B5500</td>
<td>48</td>
<td>6+S</td>
<td>39+S</td>
<td></td>
</tr>
<tr>
<td>B8500</td>
<td>48</td>
<td>10+S</td>
<td>35+S</td>
<td></td>
</tr>
<tr>
<td>CDC 36/3800</td>
<td>48</td>
<td>11</td>
<td>36</td>
<td>11</td>
</tr>
<tr>
<td>CDC 6600</td>
<td>60</td>
<td>11+S</td>
<td>48</td>
<td>11+S</td>
</tr>
<tr>
<td>GE 625/35/45</td>
<td>36,72</td>
<td>7+S</td>
<td>27+S</td>
<td>7+S</td>
</tr>
<tr>
<td>IBM 360</td>
<td>16,32</td>
<td>7</td>
<td>24</td>
<td>7</td>
</tr>
<tr>
<td>IBM 360/44</td>
<td>16,32</td>
<td>7</td>
<td>24</td>
<td>7</td>
</tr>
<tr>
<td>RCA SPECTRA 70</td>
<td>16,32</td>
<td>7</td>
<td>24</td>
<td>7</td>
</tr>
<tr>
<td>SDS SIGMA 7</td>
<td>32</td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>UNIVAC 494</td>
<td>15,30</td>
<td></td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>UNIVAC 1108</td>
<td>36</td>
<td>8</td>
<td>27+S</td>
<td>11</td>
</tr>
</tbody>
</table>

(SOFTWARE)

(-f; - ONLY)
EARLY DEVELOPMENTS

- MILITARY INFLUENCE
  - SAGE
  - L-SYSTEMS
- SHARED-DEVICE SYSTEMS
  - ASP/HASP
  - ON-LINE 1401
- MIT/CTSS
- DARTMOUTH BASIC
- IBM QUIKTRAN
BASIC ELEMENTS OF TIME-SHARING

- ON-LINE UTILIZATION
- TERMINAL INTERFACE
- ILLUSORY USE OF VIRTUAL MACHINE
- HUMAN VS. MACHINE RESPONSE TIME
TYPES OF MULTIPROGRAMMING SYSTEMS

- SPECIAL PURPOSE
  - DEDICATED MACHINE
  - FIXED PROGRAM STRUCTURE
  - HIGHLY VARIABLE DATA LOADS
  - EXAMPLES:
    - AIRLINE RESERVATIONS
    - THEATER TICKET
    - BROKERAGE

- LIMITED PURPOSE
  - DESIGNED FOR ONE LANGUAGE
  - BASIC
  - QUIKTRAN

- GENERAL PURPOSE
  - PURE MULTIPLE BATCH
  - PURE ON-LINE
  - MIXED BACKGROUND/FOREGROUND
OPERATING SYSTEM PRINCIPLES

- MUST ACCOMMODATE MULTIPROCESSING
- HANDLES MANY USERS
- HANDLES VARIED USER NEEDS
- COMPUTER UTILITY
- ALLOCATION OF ALL FACILITIES
- DEVICE-INDEPENDENCE
- SCHEDULING
- SWAPPING
- RESPONSIVENESS AND RELIABILITY

OTHER CONSIDERATIONS

- NEED FOR ON-LINE LANGUAGES
- CONVERSATIONAL/NONCONVERSATIONAL
- MIXED-MODE OPERATIONS
MAPPING OF $2^a$ ONTO $2^b$, ($b > a$).
REASONS FOR COMPACTING

- FREE LARGEST BLOCKS OF CONTIGUOUS CORE
- ALLOWS FLEXIBILITY IN CHOOSING NEXT USER
- PROVIDES CORE REQUEST/RELEASE
- TO PROVIDE A MEMORY SPACE $2^b$ WHICH ACCOMMODATES THE NAME SPACE $2^a$ ($b > a$).
- CONTROL SWITCHED BY RE-SETTING BAR.
COMPACTING FOR MEMORY RE-ALLOCATION

<table>
<thead>
<tr>
<th>UNUSED</th>
<th>UNUSED</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>F</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>
REQUIREMENTS FOR COMPACTING

- ALL PROGRAMS PRE-BOUND
- ALL PROGRAMS SELF-RELATIVE
- BASE ADDRESS REGISTER USED
- NO MOVING DURING I/O OPERATIONS
- ALL QUEUES MUST BE DRAINED
- ALL BUFFERING MUST RE-START
- NO SHARED REFERENCES
ALLOCATION PROBLEM:
NAME SPACE OF G > MEMORY SPACE

F
E
D
C
B
A

G
RESOURCE-INDEPENDENCE

- PROGRAMMER CONTROLS TIME SEQUENCE
- SYSTEM CONTROLS RESOURCE ALLOCATION
- PROGRAMMER REFERENCES NAMES
- SYSTEM TRANSFORMS NAMES TO DEVICES
- PROGRAMMER USES VIRTUAL LANGUAGE
- SYSTEM INTERPRETS VIRTUAL LANGUAGE
- PROGRAMMER SEES VIRTUAL PROCESSOR, VIRTUAL MEMORY, VIRTUAL REGISTERS
- SYSTEM ALLOCATES PHYSICAL RESOURCES TO MATCH VIRTUAL RESOURCES
SEGMENTATION WITH FIXED BLOCKS

(PROGRAMMER) 1
(ProGReMMer) 2
(PROGRAMMER) 3

P1

S1
S2
S3
S4

P2

S1
S2
S3

P3

S1
S2

P1, S1
P1, S2
P1, S3
P1, S4
P2, S1
P2, S2
P2, S3
P3, S1
P3, S2
RESOURCE ALLOCATION AND TIME-SHARING

SEGMENTATION INTO NON-CONTINUOUS MEMORY

(PROGRAMMER) -> PROCESS

STATE WORD -> MOD. ADDRESS -> SEG. WORD -> PHYS. ADDRESS

SEGMENT 1

SEGMENT 2

SEGMENT n

SEGMENT 1

SEGMENT 2

SEGMENT n
THE SEGMENTS OF A PROCESS

- EXECUTABLE SEGMENT
- READ-ONLY SEGMENT
- READ-WRITE SEGMENT
- ALTERABLE SEGMENT
- EXECUTABLE SEGMENT
- PRIVATE SEGMENT
- DESCRIPTOR SEGMENT
- LINKAGE SEGMENT

12.14
TWO PROCESSES WITH A SHARED SEGMENT
USE OF SHARED SEGMENT FOR SYSOUT

P1,S1

P2,S1

P1,S2

P1,S3/P2,S2
PAGING

- PROVIDES ADDITIONAL LEVEL OF CORE USAGE
- IMPLEMENTED BY HARDWARE
- REQUIRES SUBSTANTIAL SOFTWARE INTEGRATION
- IMPORTANT FOR ADVANCED SYSTEMS
TOPICS TO BE COVERED THIS SESSION

- REVIEW OF MULTIPROGRAMMED/MULTIPROCESSOR CONTROL PHILOSOPHY
- ORIGINS OF 'TIME-SHARING'
- HARDWARE/SYSTEMS DEVELOPMENTS FOR TIME-SHARING
- GE 645
- 360/67
- OTHER 'TIME-SHARING' SYSTEMS
TIME-SHARING CHARACTERISTICS

- TIME-SHARING IS AN OUTGROWTH OF MULTIPROGRAMMING
- TERM ASSOCIATED WITH 'INTERACTIVE' OR 'ON-LINE' COMPUTING WHERE USERS PRESENCE (OR INTERVENTION) IS REQUIRED FOR SUCCESSFUL OPERATION OF A PROGRAM
- LACK OF ON-LINE COMPONENT YIELDS SIMPLE MULTI-PROGRAMMING
- ON-LINE COMPONENT PERMITS SYSTEM TO SERVE MANY MORE ON-LINE USERS BECAUSE OF USER INTRODUCED DELAY (SO-CALLED 'THINK' TIME)
- NEEDS MECHANISM FOR MAKING PHYSICAL SPACE AVAILABLE TO USERS -- SWAPPING
CTSS SYSTEM - FUNCTIONAL DESCRIPTION

A
CTSS SUPERVISORY PROGRAM
32K

B
USERS PROGRAMS IN EXECUTION
32K

7094 I

A B C D E F G
6 TAPES PRINTER PUNCH CR

6 TAPES
DIRECT DATA CHANNEL

1302-2 DISC (9.3x10^6 WORDS)
M320 DRUM

7320A DRUM (208,608 WORDS)
SAME AS CHANNEL D

7750 TRANSMISSION CONTROL (TO USER'S CONSOLES) - 1050's MOD 35 TTs
PERTAINENT EXPERIENCE WITH CTSS

- HIGH OVERHEAD FOR SWAPPING
- 'GROWTH' OF DATA AREAS—LIST PROCESSING, ON-LINE COMPILING/ASSEMBLY
- PRACTICAL LIMIT OF 25-30 ON-LINE USERS
- GENERAL COMPUTING REQUIREMENTS
- NOTION OF COMPUTER UTILITY
APPROACHES TO PROVIDING USER ADDRESS SPACE

- EARLY ASSEMBLERS
  REGIONAL ADDRESSING
- ALGOL BLOCK STRUCTURE
- SEGMENT RELATIVE ADDRESSING
TWO COMPONENT ADDRESSING

SEGMENT SELECTS INFORMATION STRUCTURE SEGMENT
WORD SELECTS WORD WITHIN SELECTED SEGMENT
DEFINITIONS

- SEGMENT: AN OBJECT (CODE, DATA, etc.) IN USER ADDRESS SPACE

GENERALIZED ADDRESS: CONTAINS

SEGMENT #

WORD #

<table>
<thead>
<tr>
<th>SEGMENT</th>
<th>WORD</th>
</tr>
</thead>
</table>

descriptor: DEFINES AND LOCATES INFORMATION IN PHYSICAL MEMORY, A BASE ADDRESS
PAGE CONCEPT

- ORIGINS IN ATLAS SYSTEM
- FITS SWAPPING REQUIREMENT
- DEFINITION:
  UNIT OF RELOCATABLE STORAGE
ILLUSTRATE USE OF DESCRIPTORS IN PAGE TABLE

EACH DESCRIPTOR POINTS TO A BLOCK (PAGE)
645 REGISTERS

PC

PBR

X₀

AP

X₁

BP

...

...

...

...

X₇

LP

SP

A

DBR

Q
GE645 ADDRESSING

ASSOCIATIVE MEMORY

SEGMENT NR    WORD

GENERALIZED ADDRESS

DBR

(ADDRESS OF SEGMENT DESCRIPTOR)

S BASE
S BASE

(MATCH)

PHYSICAL ADDRESS

ASSOCIATIVE MEMORY

S BASE

(ADDRESS OF SEGMENT DESCRIPTOR)

SEGMENT NR

WORD

SEGMENT BASE?

No

Yes

(PAGE ID)

SEG. DESC.

PAGE TABLE BASE

PAGED SEGMENT?

No

SEGMENT BASE

Yes

PAGE TABLE BASE
645 INSTRUCTION ADDRESSING

<table>
<thead>
<tr>
<th>SEGMENT&lt;sub&gt;NR&lt;/sub&gt;</th>
<th>WORD&lt;sub&gt;NR&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENERALIZED ADDRESS</td>
<td></td>
</tr>
<tr>
<td>PBR</td>
<td>PC</td>
</tr>
</tbody>
</table>

- DBR
- SELECTS SEGMENT DESCRIPTOR
- SEG. DESCRIPTOR
- PHYSICAL ADDRESS

13.12
645 ADDRESSING CHARACTERISTICS

- INFORMATION STRUCTURE MAY BE
  \[2^{18}\] SEGMENTS
  EACH SEGMENT MAY BE
  \[2^{18}\] WORDS.

- PAGES
  64 OR 1024 WORDS
I/O CONTROL - GE 645

CPU

MEMORY MODULE

MEMORY MODULE

MEMORY MODULE

GIOC

GIOC

DISC CONTROLLER

DISC CONTROLLER

DISC

DISC
SYSTEM DEVELOPMENT FOR TIME-SHARING

GIOC - GE645
PRIMARY STORAGE

COMMON CONTROL

ADAPTORS

ADAPTOR

ADAPTOR CHANNEL

ADAPTOR CHANNEL

ADAPTOR CHANNEL

DEVICES OR TERMINALS
SEGMENT-PAGE ADDRESSING

ADDRESS IN USER ADDRESS-SPACE

STR

(Segment Table Base)

Page Table Base

Segment Table

Page Base

Page Table

Page
360/67 ADDRESSING

(24) GENERAL REGISTERS

Rx [OP R1 X2 B2 D2]

(24) ADDRESS ADDER

(24) NORMAL 360 ADDRESS FORMATION

(12) 360/67 INTERPRETATION

SEGMENT (4) PAGE (8) BYTE (12)

ASSOCIATIVE MEMORY

SEGMENT PAGE PHYSICAL ADDRESS (12)

(HIGH ORDER (12) LOW ORDER (12))

MATCH)

(LOADED INTO ASSOCIATIVE REGISTER WITH SEG/PAGE)

STR (24)

PAGE TABLE BASE

SEGMENT TABLE

PAGE TABLE

platzhalter
360/67 WITH PARTITIONING SWITCHES
PAGING ADVANTAGES AND DISADVANTAGES

- PERMITS ARBITRARY ALLOCATION OF STORAGE IN SMALL BLOCKS
- DEFERS BINDING UNTIL EXECUTION TIME PERMITS ALLOCATION AND EXECUTION OF FRAGMENTS OF PROGRAMS
- COUPLED WITH OPERATING SYSTEM, PERMITS EACH USER TO HAVE EXTREMELY LARGE ADDRESS SPACE
- NOT ALL PROGRAMS REQUIRE TREATMENT AS ABOVE
- EXPENSIVE IN TIME AND MONEY FOR MANY APPLICATIONS
- THERE ARE OTHER WAYS TO ACHIEVE SAME ENDS
OTHER MACHINES ORIENTED TO TIME-SHARING

- SDS 940
- SDS SIGMA 7
- CDC 3500
- PDP 10
SUMMARY OF PERTAINENT ADDRESSING CONCEPTS

SEGMENTS - COMPONENT OF USER ADDRESS SPACE

PAGE COMPONENT OF PHYSICAL ADDRESS SPACE

PAGING: MAPS SEGMENTS (USER ADDRESS SPACE) INTO PAGES (PHYSICAL ADDRESS SPACE)
SOFTWARE FOR TIME-SHARING: MULTICS

MOTIVATION

- MULTIPLE INFORMATION AND COMPUTING SERVICE
- COMPUTER UTILITY

HARDWARE

- TWO-LEVEL ADDRESSING
- ONE-LEVEL STORE
- SEGMENTATION BY USER
- PAGING BY SYSTEM

SOFTWARE

- SYMBOLIC SEGMENT REFERENCES
- RECURSIVE PROCEDURES
- LOCATION-INDEPENDENCE
- PRIVATE STACK FOR TEMPORARY STORAGE
- FILE SYSTEM
  - SYMBOLIC
  - ACCESS-CONTROLLED
SOFTWARE FOR TIME-SHARING: MULTICS

VIRTUAL MEMORY OF A MULTICS PROCESS

DIRECTORY STRUCTURE

VIRTUAL MEMORY

SEGMENTS
THE GENERALIZED ADDRESS

<table>
<thead>
<tr>
<th>SEGMENT NUMBER</th>
<th>WORD NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SOFTWARE FOR TIME-SHARING: MULTICS

PROCESSOR REGISTERS FOR ADDRESS FORMATION

<table>
<thead>
<tr>
<th>PC</th>
<th>PBR</th>
</tr>
</thead>
<tbody>
<tr>
<td>X0</td>
<td>AP</td>
</tr>
<tr>
<td>X1</td>
<td>BP</td>
</tr>
<tr>
<td></td>
<td>LP</td>
</tr>
<tr>
<td>X7</td>
<td>SP</td>
</tr>
</tbody>
</table>

INSTRUCTION FORMAT

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>EXTERNAL FLAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEGMENT TAG</td>
<td>OPERATION CODE</td>
</tr>
</tbody>
</table>

14.4
ADDRESS FORMATION FOR INSTRUCTION FETCH

GENERALIZED ADDRESS

SEGMENT NUMBER  WORD NUMBER

PC  PBR

ADDRESS FORMATION FOR DATA ACCESS

GENERALIZED ADDRESS

SEGMENT NUMBER  WORD NUMBER

SEGMENT NUMBER  WORD NUMBER

BASE REGISTER

INDEX REG.

ADDRESS  OPR  1
INTERPRETATION OF WORD PAIR AS INDIRECT ADDRESS

GENERALIZED ADDRESS

<table>
<thead>
<tr>
<th>SEGMENT NUMBER</th>
<th>WORD NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEGMENT NUMBER</td>
<td>-----</td>
</tr>
<tr>
<td>WORD NUMBER</td>
<td>-----</td>
</tr>
</tbody>
</table>
ADDRESSING BY GENERALIZED ADDRESS

<table>
<thead>
<tr>
<th>SEGMENT NUMBER</th>
<th>WORD NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
</tr>
</tbody>
</table>

DESRIPTOR SEGMENT

INFORMATION SEGMENT
ADDRESS FORMATION FOR AN UN-PAGED SEGMENT

1. DBR
2. SEGMENT NUMBER + WORD NUMBER → GENERALIZED ADDRESS
3. GENERALIZED ADDRESS + SEGMENT ADDRESS → PHYSICAL LOCATION
4. DESCRIPTOR SEGMENT: N A/C

SOFTWARE FOR TIME-SHARING: MULTICS
ADDRESS FORMATION FOR A PAGED SEGMENT

DBR

SEGMENT PAGE TABLE BASE

SEGMENT DESCRIPTOR

SEGMENT NUMBER PAGE NO. LINE NO.

PAGE LOCATION

PAGE DESCRIPTOR

PHYSICAL LOCATION

14.9
AN INTERSEGMENT REFERENCE BY PROCEDURE P

P

D

x
LINKAGE OF P TO D X FOR PROCESS α

* INDICATES INDIRECT ADDRESSING

A)

\[ \begin{array}{c|c}
& ft \\
\hline
\text{MODE} & \\
\end{array} \]

POINTER TO< D> [X]

B)

\[ \begin{array}{c|c}
D#_\alpha & \text{its} \\
\hline
X & \text{MODE} \\
\end{array} \]

STATES OF THE LINK DATA
ADDRESSING THE LINK DATA

LINKAGE SECTION FOR P

L_p  |  k  |  OPR  |  1  |  *
PROCEDURE P IN PROCESS $\alpha$ BEFORE LINKAGE

PROCEDURE P IN PROCESS $\alpha$ AFTER FIRST EXECUTION
REFERENCE TO COMMON DATA SEGMENT BY TWO PROCEDURES

BEFORE & AFTER LINKAGE

BEFORE ONLY

AFTER ONLY
LINKAGE MECHANISM FOR PROCEDURE ENTRY

CALL <Q>[e]

CONTROL FLOW
INDIRECT ADDRESS FLOW
STATUS OF A PROCESS

- RUNNING
- READY
- BLOCKED
AN EXAMPLE OF A HIERARCHY
THE SAME HIERARCHY WITH LINKS ADDED

DIRECTORY MANIPULATION

1. SUPPOSE CURRENT WORKING DIRECTORY IS 4 (PATHNAME H)

2. THE COMMAND CHANGE DIRECTORY : C WILL ALTER THE WORKING DIRECTORY TO 1 (PATHNAME H: C)

3. A SUBSEQUENT REFERENCE TO :* : I WILL THEN INDICATE BRANCH 5.
ACCESS CONTROL

- USER ACCESS CONTROL LIST
- MODE ATTRIBUTES:

<table>
<thead>
<tr>
<th>MODE</th>
<th>DIRECTORY BRANCH</th>
<th>NON-DIRECTORY BRANCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ:</td>
<td>READ AVAIL. CONTENTS</td>
<td>READ FILE</td>
</tr>
<tr>
<td>WRITE:</td>
<td>ALTER EXISTING ENTRIES</td>
<td>WRITE FILE</td>
</tr>
<tr>
<td>EXECUTE:</td>
<td>SEARCH THE DIRECTORY</td>
<td>EXECUTE PROCEDURE</td>
</tr>
<tr>
<td>APPEND:</td>
<td>ADD NEW ENTRIES</td>
<td>WRITE AT E.O.F.</td>
</tr>
</tbody>
</table>

- THE TRAP ATTRIBUTE
  - MONITORS FILE USAGE
  - RESTRICTS ACCESS
  - DYNAMIC REFERENCE CONTROL
THE BASIC FILE SYSTEM
SEGMENT MANAGEMENT

- Maintains record of all known segments (S, N, T.)
  
  Active: If page table in core (S, S, T.)
  
  Inactive: If page table not in core
  
  Calls linker for first-time reference
  
  If not in SNT, locate segment, assign segment number, update SNT, open file, create SST entry, set up page table and segment descriptor; then
  
  Return segment number to calling procedure
  
  If in SNT but inactive, activate
  
  Other functions:
  
  Release, reassign, verify, create, terminate
SOFTWARE FOR TIME-SHARING: MULTICS

SEARCH MODULE

- CALLED SEGMENT MANAGEMENT
- USES FILE COORDINATOR
- LOCATES SPECIFIC BRANCH IN USER'S HIERARCHY

FILE COORDINATOR

- BASIC WORKING DIRECTORY ENTRY MANIPULATION
- INTERFACES WITH ACCESS CONTROL FOR PERMISSION
- KEEP TREE NAME OF WORKING DIRECTORY IN WDT
- FUNCTIONS:
  - CREATE, DELETE, RENAME AN ENTRY
  - STATUS OF AN ENTRY
  - CHANGE ACCESS CONTROL FOR A BRANCH
  - CHANGE WORKING DIRECTORY
SOFTWARE FOR TIME-SHARING: MULTICS

DIRECTORY MANAGEMENT

- Searches for a single directory by tree name
- May call segment management to get segment number
- May be re-called by segment management
- Recursion may reach to root of tree

FILE CONTROL MODULE

- Opens files for segment management
- Makes entry in active file table (AFT)
- Return AFT pointer
- Gets permission from access control module
- May block process on incompatible request

14.23
SOFTWARE FOR TIME-SHARING: MULTICS

ACCESS CONTROL MODULE

- CHECKS DIRECTORY, RETURN EFFECTIVE MODE
- FOR TRAP MODE, PASSES CONTROL TO INDICATED PROCEDURE FOR EFFECTIVE MODE DETERMINATION

PAGE MARKER

- PERIODICALLY INTERRUPTS
- RESETS PAGE USE BITS
- PUTS SELDOM—USED PAGE DATA IN PAGE OUT TABLE (POT)
**PAGE MANAGEMENT MODULE**

- ENTERED BY MISSING PAGE FAULT
- ASSIGNS FREE PAGE FROM AVAILABLE SPACE OR POT
- FOR NEW PAGE, POINTER FROM PAGE TABLE TO SEGMENT STATUS
  TABLE USED TO GET POINTER TO ACTIVE FILE
- POINTER PASSED TO I/O QUEUE MANAGEMENT TO READ PAGE
SOFTWARE FOR TIME-SHARING
TSS/360

TIME SLICING AMONG THREE TASKS IN TSS/360

Points
A B C D E F G H I

Time Slices
1 2 3 4 5 6 7

- Task Active
- Task Waiting for I/O
- Task Inactive
PRIVATE CODE AND SHARED CODE
HOW CALLS WORK IN A BEGIN I/O OPERATION EXAMPLE

PROBLEM PROGRAMS FORTRAN, ASSEMBLER
SVC SUPERVISOR CALL INSTRUCTION
SERVICE PROGRAMS ACCESS METHODS, COMMAND LANGUAGE
LOCAL MACRO
RESIDENT SUPERVISOR
SIO PRIVILEGED INSTRUCTION
HARDWARE MICRO-PROGRAM


PROBLEM
PROBLEM
YES
YES
YES
NONE

SUPERVISOR
YES
YES
YES
PROTECT KEY

SUPERVISOR
NO
NO
NO

NOT IN VIRTUAL MEMORY

READ-ONLY

PROTECTION LEVELS

SOFTWARE FOR TIME-SHARING
TSS/360
SOFTWARE FOR TIME-SHARING
TSS/360

NONCONVERSATIONAL TASK
INITIATED BY PRINT COMMAND

User

LOGON

(PRINT Data Set A)

LOGOFF

SYSIN
Task 1

TSS/360

Data Set A

SYSOUT
Task 1

Printer

Data Set A

Task 2

SYMON
Task 1
SOFTWARE FOR TIME-SHARING
TSS/360

NONCONVERSATIONAL TASK
INITIATED BY EXECUTE COMMAND

User

Task 1
Conversational

LOGON
EXECUTE (Procedure A)
LOGOFF

SYSIN
Task 1

SYSOUT
Task 1

Direct-Access Device

Procedure A
LOGON

LOGOFF

Task 2
Nonconversational

SYSIN
Task 2

SYSOUT
Task 2

Intermediate Storage

TSS/360

Printer

14.31
CONVERTING A CONVERSATIONAL TASK TO NONCONVERSATIONAL MODE USING THE BACKGROUND COMMAND
TIME SHARING SYSTEM/360 DATA MANAGEMENT FACILITIES

**DATA SET MANAGEMENT**

- Cataloging
  - System catalog
  - Cataloging facilities (including CATALOG and DELETE commands)
- Sharing
  - PERMIT command
  - SHARE command
- Manipulation
  - MODIFY command
  - COPY DATA SET command
  - ERASE command
- Definition
  - DEFINE DATA command
  - CALL DATA DEFINITION command
  - RELEASE command
  - SECURE command

**PROBLEM PROGRAM I/O**

- Prestore input data in system
  - DATA command -- by user
  - READ CARDS command -- by operator
  - READ TAPE command -- by operator
- Obtain input data and generate output data during program execution
  - Conventional I/O facilities, using I/O statements in source program
  - Dynamic I/O facilities, using program checkout commands and statements, and special source language statements
- Transfer data from system storage to standard I/O devices
  - PRINT command
  - PUNCH command
  - WRITE TAPE command
FULLY AND PARTIALLY QUALIFIED NAMES

Data Set Name Structure

1. A
2. B
3. C

Data Set A.A
Data Set A.B.A
Data Set A.B.B
Data Set A.B.C
Data Set A.C
SYSTEM CATALOG CONCEPT
TYPICAL VIRTUAL INDEX SEQUENTIAL DATA SET
VIRTUAL PARTITIONED DATA SET

Partitional Organization Directory—1 Page

Entry for Member ARCT

Entry for Member COS

Entry for Member SQRT

Member SQRT (First Part)

Member SQRT (Continued)

Member ARCT (First Part)

Member ARCT (Continued)

Member COS
SOFTWARE FOR TIME-SHARING
TSS/360

SHARING OF CATALOGED DATA SETS

<table>
<thead>
<tr>
<th>Share</th>
<th>ENG, CHEM, NOTAR, TEST1</th>
<th>RKP100, ENG, PHYSICS, COMAR, TEST2</th>
</tr>
</thead>
</table>

- Share: Sharer's Reference to Data Set
- Owner's Identification of Data Set
- Data Set's Owner

JMC200

ENG, CHEM, NOTAR, TEST1

JMC's User Catalog

ENG

CHEM

NOTAR

TEST1

RKP100

ENG, PHYSICS, COMAR, TEST2

RKP's User Catalog

ENG

PHYSICS

COMAR

TEST1

TEST2

Volume Identification

Volume Label

VTOC

Cylinder 0-Track 0

RKP100, ENG, PHYSICS, COMAR, TEST2

Data Set Beginning Address

14.38
FLOW OF INFORMATION TO AND FROM A DATA CONTROL BLOCK

Note: Circled numbers 1-3 indicate order of sampling sources for inputs to DCB. Circled numbers 4-6 indicate sources of data set label information and order they are used when output data sets are opened. Boxed numbers 1-4 show priorities of sources sampled for inputs.
DATA SET IDENTIFICATION, FORTRAN-WRITTEN PROGRAMS

READ or WRITE Statement
(data set reference number xx
FT xx Fyyy
DEFINE DATA command
DSNAME = dsname

dsname in data set label
DATA SET

DATA SET IDENTIFICATION, ASSEMBLER LANGUAGE PROGRAM

Macro Instructions
(GET, PUT, READ, WRITE, etc)
dcb address
Data Control Block
dname

DEFINE DATA command
DSNAME = dsname
dname in data set label
DATA SET

14. 40
## SUMMARY OF DATA MANAGEMENT SYSTEM MACRO INSTRUCTIONS AND DATA SET ORGANIZATIONS

### General Service Macro Instructions
Applicable in All Access Methods

<table>
<thead>
<tr>
<th>DCB</th>
<th>DCBD</th>
<th>OPEN</th>
<th>CLOSE</th>
</tr>
</thead>
</table>

### Virtual Sequential Macro Instructions
- **GET**
- **PUT**
- **PUTX**
- **SETL**

### Virtual Index Sequential Macro Instructions
- **GET**
- **PUT**
- **READ**
- **WRITE**
- **SETL**
- **ESETL**
- **DELREC**
- **RELEX**

### Virtual Partitioned Macro Instructions
- **FIND**
- **STOW**

### Basic Sequential Macro Instructions
- **GETPOOL**
- **FREEPOOL**
- **GETBUF**
- **FREEBUF**
- **FEOV**
- **CNIKL**
- **PRTOV**
- **READ**
- **WRITE**
- **CHECK**
- **NOTE**
- **POINT**
- **BSP**
- **CLOSE (TYPE -1)**
- **DQDECB**

### Input/Output Request Facility Macro Instructions
- **VCCW**
- **IOREQ**
- **CHECK**

### Virtual Sequential Data Set or Virtual Sequential Member of a Partitioned Data Set

### Virtual Index Sequential Data Set or Virtual Index Sequential Member of a Partitioned Data Set

### Virtual Partitioned Data Set, with Virtual Sequential or Virtual Index Sequential Members or a Mixture of Both

### Sequential Data Set, Usually One with Unblocked Records

### Device Oriented
FORMAT OF AN OBJECT PROGRAM MODULE

<table>
<thead>
<tr>
<th>Program Module Dictionary</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMD Header</td>
</tr>
<tr>
<td>Control Section 1</td>
</tr>
<tr>
<td>Dictionary</td>
</tr>
<tr>
<td>Control Section 2</td>
</tr>
<tr>
<td>Dictionary</td>
</tr>
<tr>
<td>Control Section 3</td>
</tr>
<tr>
<td>Dictionary</td>
</tr>
<tr>
<td>Control Section η</td>
</tr>
<tr>
<td>Dictionary</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction and/or Data (Hexadecimal)</td>
</tr>
<tr>
<td>Control Section 1</td>
</tr>
<tr>
<td>Control Section 2</td>
</tr>
<tr>
<td>Control Section 3</td>
</tr>
<tr>
<td>Control Section η</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Internal Symbol Dictionary</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Optional)</td>
</tr>
</tbody>
</table>

14. 42
ATTRIBUTES OF CONTROL SECTIONS

- READONLY
- PUBLIC
- PSECT
- COM
- PRVLGD
- VARIABLE
V- AND R-VALUES OF EXTERNAL SYMBOLS

V(M) R(M) V(X) R(X) V(Z) R(Z) V(Y) R(Y)

X CSECT
W (Standard Entry Point)
Z (Deferred Entry Point)

Y PSECT
ENTRY Z
END (W)
SHARING A MODULE

Module A
(of task 1)

X
CSECT
Y
PSECT

M

Module B
(of task 2)

14.45
PROGRAM WITH IMPLICIT AND EXPLICIT LINKAGES

Implicit Call

Explicit Call
OBJECT PROGRAM MODULE COMBINATION

Task Libraries

Object Program Modules

Dynamic Loader

User's Virtual Storage

User

Optional Linkage Editing

Edited Object Program Module

Object Program Modules

DEFINE DATA RUN Program

DEFINE DATA RUN LINK Parameters Control Statements

Linkage Editor

Listing Data Set

User Catalog

Listing Data Set/Control Statement Data Set

Input Data Sets

Output Data Sets

Explicit and Implicit Linkage Requirements During Execution

Objects Program Modules

User
A REENTERABLE ROUTINE THAT REQUESTS ITS OWN TEMPORARY STORAGE
INSIGHTS INTO

• MACHINE ORGANIZATION
• PROGRAMMING LANGUAGES
• PROGRAMMING SYSTEMS

BY MEANS OF

• CONCEPTUAL FRAMEWORK
• CASE STUDIES
## BASIC DEFINITIONS

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ALGORITHM</strong></td>
<td>A rule for computing the solution to a problem or class of problems in a finite number of steps.</td>
</tr>
<tr>
<td><strong>PROGRAM</strong></td>
<td>Representation of an algorithm in some programming language.</td>
</tr>
<tr>
<td><strong>COMPUTER</strong></td>
<td>Mechanical device for program execution.</td>
</tr>
<tr>
<td><strong>COMPILER</strong> (TRANSLATOR)</td>
<td>Program for translating from one programming language to another.</td>
</tr>
<tr>
<td><strong>SOURCE LANGUAGE</strong></td>
<td>Programming language in which programs are specified by the programmer or programming language which serves as input to a compiler.</td>
</tr>
<tr>
<td><strong>TARGET LANGUAGE</strong></td>
<td>Programming language which serves as output from a compiler.</td>
</tr>
<tr>
<td><strong>ASSEMBLER</strong></td>
<td>Special case of a compiler when translation from the source language to the target language involved mainly transliteration.</td>
</tr>
<tr>
<td><strong>PROGRAMMING SYSTEM</strong></td>
<td>A set of programs for a computer which allows sequences of user programs to be executed without manual intervention. The term programming system sometimes denotes the hardware of the computer system together with the set of programs that constitute the interface between the hardware and the user.</td>
</tr>
</tbody>
</table>
CONCEPTS OF A FUNCTION

\[ X \xrightarrow{F} Y = F(X) \]

MATHEMATICAL CONCEPT OF A FUNCTION

COMPUTATIONAL CONCEPT OF A FUNCTION
REPRESENTATIONS OF A FUNCTION

A REPRESENTATION OF A FUNCTION \( F \) TOGETHER WITH ITS DATA \( X \) CONSTITUTES AN INFORMATION STRUCTURE. A FINITE COMPUTATION CAN BE CHARACTERIZED BY AN INITIAL INFORMATION STRUCTURE \( I_0 \), AND BY THE SEQUENCE OF INFORMATION STRUCTURES \( I_1; I_2 \ldots I_N \) GENERATED FROM \( I_0 \) BY THE EXECUTION OF INSTRUCTIONS. \( I_0 \) IS SAID TO BE THE INITIAL REPRESENTATION AND \( I_N \) IS SAID TO BE THE FINAL REPRESENTATION. AN INFORMATION STRUCTURE \( I_j \) WHICH CAPTURES THE COMPLETE STATE OF THE COMPUTATION AT A GIVEN POINT IN ITS LIFETIME IS SAID TO BE AN INSTANTANEOUS DESCRIPTION.
FUNCTIONAL COMPONENTS OF A COMPUTER

PROCESSING UNIT

INSTRUCTION POINTER

0 1 K N-1

A SIMPLE COMPUTER

AUXILIARY MEMORY

INPUT UNIT

MEMORY

OUTPUT UNIT

PROCESSING UNIT

FUNCTIONAL COMPONENTS OF A COMPUTER
TRANSLATION, COMPILATION AND LOADING

SOURCE LANGUAGE → COMPILER → TARGET LANGUAGE

SOURCE LANGUAGE COMPONENTS → COMPILER → INTERMEDIATE LANGUAGE COMPONENTS

INTERMEDIATE LANGUAGE COMPONENTS → LOADER → MACHINE LANGUAGE PROGRAM
REQUIRED PROPERTIES OF INTERMEDIATE LANGUAGE (COMPILER)

- PROGRAM REPRESENTATION INDEPENDENT OF MACHINE STORAGE LOCATIONS.

- PROVISION FOR CROSS-REFERENCING BETWEEN PROGRAM COMPONENTS.

- TRANSLATION TO PURE MACHINE LANGUAGE AS EFFICIENT AS POSSIBLE.
PROGRAM STRUCTURE FOR FORTRAN

- MAIN PROGRAM

- SUBROUTINES

- COMMON DATA BLOCKS

PRINCIPAL COMPONENTS OF A FORTRAN PROGRAM UNIT
FUNCTIONAL COMPONENTS OF A PROGRAM

- A PROGRAM PART P WHICH SPECIFIES THE PROGRAM TO BE EXECUTED.

- A DATA PART D WHICH SPECIFIES THE DATA FOR THE PROGRAM.

- A STATEWORD W WHICH CONTAINS INFORMATION IN THE PROCESSING UNIT OF AN ACTUAL COMPUTER, INCLUDING AN INSTRUCTION POINTER WHICH POINTS TO THE NEXT STATEMENT OR SUBEXPRESSION TO BE EXECUTED.

LOGICAL PROGRAM STRUCTURE
DEFINITIONS OF FUNCTIONS

• ACTIVATION RECORD

• REENTRANT FUNCTIONS

• RECURSIVE FUNCTIONS
SEQUENCE OF FUNCTIONAL COMPONENTS

PROGRAM STRUCTURE

STACK WHEN EXECUTION IS IN D

STACK WHEN EXECUTION IS IN E

PROGRAM STRUCTURE AND ACTIVATION RECORD STACK
PROGRAM EXECUTION

- LOGICAL STRUCTURE

- PHYSICAL STRUCTURE

- MACHINE ORGANIZATION
COMMUNICATION BETWEEN FUNCTION MODULES

- Symbolic Cross References
- Transfer Vectors
- Load Time Linkage
- One and Two-Stage Indirect Addressing
- Incremental Linkage
ONE AND TWO-STAGE INDIRECT ADDRESSING

USE AND DEFINITION TABLES FOR PROGRAMS IN THE INTERMEDIATE LANGUAGE.

<table>
<thead>
<tr>
<th>EXTERNALLY DEFINED SYMBOL</th>
<th>SYMBOL BEING DEFINED</th>
</tr>
</thead>
<tbody>
<tr>
<td>USE 1</td>
<td>USE 2</td>
</tr>
<tr>
<td>SYMBOL DEFINITION</td>
<td></td>
</tr>
</tbody>
</table>

USE-TABLE ENTRY          DEFINITION-TABLE ENTRY

<table>
<thead>
<tr>
<th>USE TABLE</th>
<th>DEFINITION TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BODY OF PROGRAM UNIT</td>
<td></td>
</tr>
</tbody>
</table>

INDIRECT ADDRESSING OF STORAGE-MAPPING TABLE

TRANSFER VECTOR

SYMBOLIC ENTRY
PRIOR TO LOADING.
LINK TO EXTERNAL VALUE DURING EXECUTION

ENTRY FOR X
FIRST USE OF X
SECOND USE OF X
THIRD USE OF X

POINTER ESTABLISHED DURING TRANSLATION

EXECUTION-TIME STORAGE-MAPPING-TABLE ENTRY

USE-TABLE ENTRIES IN THREE PROGRAM UNITS
USES OF EXTERNALLY DEFINED SYMBOL

USES IN P1
USES IN P2
USES IN P3
STARTING POINT FOR THE STUDY OF PROGRAMMING

ALGORITHMS
COMPUTERS
INFORMATION STRUCTURES

COMPUTER SCIENCE CAN BE DEFINED AS THE STUDY OF REPRESENTATION AND TRANSFORMATION OF INFORMATION STRUCTURES.
INFORMATION STRUCTURES

ALPHABET T

INFORMATION STRUCTURE OVER T IS A SYMBOL STRING OVER T

SUBSTRUCTURE IMPOSED ON STRINGS BY A GRAMMAR

BEGIN REAL X; X: = 3 + 4 x 5 END

DECLARATION EXPRESSION

STATEMENT

BLOCK

PROGRAMMING LANGUAGE - SET OF INFORMATION STRUCTURES

SYNTAX - SPECIFIES REPRESENTATION

SEMANTICS - SPECIFIES TRANSFORMATION
INFORMATION STRUCTURE MODELS

\((I, F)\)  \(I\) is set of information structures

\(F\) is set of transformations

\(I\) - syntactic component - specified by syntax

\(F\) - semantic component - specified by semantics

computation \( I_0 \xrightarrow{f} I_1 \xrightarrow{f} I_2 \ldots \xrightarrow{f} I_n \)

\( I_0 \in I \) initial representation

\( I_i \) intermediate representations - instantaneous descriptions

\( I_n \) final representation - no elements of \( f \) are applicable

Closure of \( I \) - set of all information structures which can be generated from \( I \) by finite sequences of \( f \).
INFORMATION STRUCTURE MODEL FOR COMPUTERS

STORAGE STRUCTURES

PRIMITIVE INSTRUCTIONS

Principal information components

Processing unit component \( PU \)
Memory component \( M \)
Instruction pointer component \( PTR \)

Syntax: \( I \rightarrow PU \rightarrow M \rightarrow PTR \)
\( PU \rightarrow AC \rightarrow MQ \rightarrow BITS \)

etc

Semantics: Specify instructions in terms of which information fields they transform.

Recognition Phase

Transformation Phase

Interpretation step: \( \text{if } p_1 \text{ then } A_1 \text{ else if } p_2 \text{ then } A_2 \ldots \text{ else if } p_n \text{ then } A_n. \)
INFORMATION STRUCTURE MODEL FOR PROGRAMMING LANGUAGES

Stateword Component \( W \)
Program Component \( P \)
Data Component \( D \)

\( W \) component is usually of fixed size

\( P \) consists of interacting function modules
reentrant function modules

Programming languages may be characterized by the structure of their \( D \) component.

FORTRAN - All information fields of the \( D \) component are determined prior to execution.

ALGOL - The \( D \) component is a stack with respect to creation and deletion of information structures.

List Processing Languages - More flexible creation and deletion.
BASIC CONCEPTS IN PROGRAMMING LANGUAGE

FORTRAN

Function module - subroutine or main program

- Program Part
- Working Space
- Data Part

One-to-one correspondence between program and data components of function module.
Complete program - set of interacting function modules and COMMON data blocks.

Program with two function modules and a COMMON data block.
COMMUNICATION BETWEEN FUNCTION MODULES

SIZE OF FUNCTION MODULES KNOWN AT TRANSLATION TIME

RELATIVE ADDRESSING WITHIN FUNCTION MODULE

RELATIVE ADDRESS FOR COMMON DATA BLOCKS

SYMBOLIC SUBROUTINE REFERENCES

PARAMETERS – RELATIVE ADDRESSING WITH RESPECT TO POINT OF CALL

TSR S, 4
A1
A2
A3

A1, A2, A3 ARE ADDRESSES OF PARAMETER VALUES

ACTUAL PARAMETER EXPRESSION IS EVALUATED PRIOR TO SUBROUTINE ENTRY

CALL BY REFERENCE
ALGOL

A PROGRAM CONSISTS OF A SINGLE FUNCTION MODULE CALLED A BLOCK WHICH MAY HAVE NESTED FUNCTION MODULES.

BEGIN

DECLARATIONS  

REAL X;  

PROCEDURE P(X) BODY

STATEMENTS

X:=X+1;  

NESTED BLOCK

END

DECLARED INFORMATION STRUCTURES ARE CREATED ON ENTRY TO BLOCK AND DELETED ON EXIT FROM BLOCK.

NESTED FUNCTION MODULES — ACTIVATION RECORD STACK

FIXED PROGRAM PART  EXECUTION AT P  EXECUTION AT Q

STATIC AND DYNAMIC NESTING OF FUNCTION MODULES
PROCEDURE CALLS ARE IMPLICITLY NESTED
OWN VARIABLES — ENDURE BETWEEN ACTIVATIONS
INFORMATION STRUCTURE MODEL FOR ALGOL

FIXED PROGRAM COMPONENT
STATEWORD COMPONENT
STACK COMPONENT
INPUT COMPONENT
OUTPUT COMPONENT
OWN VARIABLE COMPONENT

\[ I = (P, W, S, IN, OUT, X) \]

SPECIFY TRANSFORMATION F IN TERMS OF HOW THEY AFFECT INFORMATION COMPONENTS

EMPHASIZE CREATION AND DELETION OF INFORMATION FIELDS

CREATION OF ACTIVATION RECORDS ON ENTRY TO FUNCTION MODULES — DELETION ON EXIT FROM FUNCTION MODULES.

CREATION OF TEMPORILY INFORMATION FIELDS DURING EXPRESSION EVALUATION.

ASSIGNMENT STATEMENT MAY MODIFY AN INFORMATION FIELD IN THE INTERIOR OF THE STACK.
INTERPRETATION VERSUS COMPILATION

Compilation is a transformation from one initial representation to another.

\[
\begin{align*}
I_0 & \rightarrow I_1 \rightarrow I_2 \cdots \rightarrow I_N \\
I'_0 & \rightarrow I'_1 \rightarrow I'_2 \cdots \rightarrow I'_M \\
\end{align*}
\]

Interpretation process is insensitive to compilations which preserve the identity of operators and operands and the order in which operators are applied to operands.

Interpretation is more relevant to machine organization than compilation.

Compilers constitute an interesting class of computations to study but tell us little about the semantics of programming languages being compiled.
MODELLING LANGUAGES

A LANGUAGE FOR SPECIFYING INFORMATION STRUCTURE MODELS IS CALLED A MODELLING LANGUAGE.

A MODELLING LANGUAGE MUST CONTAIN SYNTACTIC SPECIFICATION FACILITIES FOR SPECIFYING THE COMPONENT OF INFORMATION STRUCTURE MODELS, AND FLEXIBLE FACILITIES FOR SPECIFYING CREATIONS, DELETION AND MODIFICATION OF INFORMATION STRUCTURES.

THERE ARE SIMILARITIES BETWEEN MODELLING LANGUAGES AND COMPILER–COMPILER LANGUAGES, BUT MODELLING LANGUAGES ARE CONCERNED WITH INTERPRETATION RATHER THAN WITH COMPILATION.

A SPECIFICATION OF AN INFORMATION STRUCTURE MODEL IN A MODELLING LANGUAGE WILL BE CALLED A SYNTAX DIRECTED INTERPRETER.

AN IMPLEMENTATION OF A MODELLING LANGUAGE WILL BE CALLED AN INTERPRETER–INTERPRETER SINCE IT IS AN INTERPRETER WHICH EXECUTES INTERPRETERS.
BINDING TIME

DECLARATIVE ACTION — REAL N;

IMPERATIVE ACTION — X := 5;

DECLARATIVE ATTRIBUTES REMAIN INVARIANT DURING LIFETIME OF STRUCTURE.

IMPERATIVE ATTRIBUTES MAY BE MODIFIED DURING EXECUTION.

BINDING TIME OF AN ATTRIBUTE

TYPE IS BOUND AT DECLARATION TIME

VALUE IS BOUND AT ASSIGNMENT TIME

FORTRAN — ALL DATA STRUCTURES ARE CREATED (BOUND) PRIOR TO EXECUTION.

ALGOL — DATA STRUCTURES MAY BE NESTED ON BLOCK ENTRY.

PL/I — TEMPLATES FOR NEW DATA STRUCTURES MAY BE DECLARED.
EXAMPLES OF BINDING

Compilation – early binding of target language

Interpretation – late binding of target language

Macros – binding of users body by substitution

Procedures – no binding by physical substitution

Parameter call by value – bind parameter at time of entry to procedure

Parameter call by name – bind parameter value when it is used in the body of the procedure.

Parameter call by reference – bind parameter address at the time of entry to the procedure

Early binding – greater efficiency

Late binding – greater flexibility
SIDES EFFECTS

When does difference in binding strategy yield different results

Strategy A - bind value V at time T₁
Strategy B - bind value V at time T₂

Different result if value of V changes between T₁ and T₂

Example - call by value - T₁ is procedure entry time - call by name - T₂ is parameter use time

Difference in result if parameter value can be changed between procedure entry and parameter use

Procedures which may change values of external parameters during execution are said to have side effects.
OBJECTIVES

• OBJECTIVES — TO DEVELOP INSIGHT AND UNDERSTANDING OF THE STRUCTURE OF THE PROGRAMMING LANGUAGES.

• START WITH A DISCUSSION OF ALGOL 60 — COMMUNICATIONS OF THE ACM JANUARY 1963.

• DEVELOPED AS AN INTERNATIONAL ALGEBRAIC LANGUAGE.

• USED AS A LANGUAGE FOR THE COMMUNICATION OF ALGORITHMS — ALGORITHMS SECTION OF THE COMMUNICATIONS OF THE ACM.

• NOT AS WIDELY USED FOR PRACTICAL PROGRAMMING AS FORTRAN.

• BUT HAS A MORE INTERESTING STRUCTURE THAN FORTRAN.

• PRIME PURPOSE IS NOT TO TEACH ALGOL PROGRAMMING BUT TO DEVELOP A MODEL FOR THE STUDY OF PROGRAMMING LANGUAGES.

• THE CONCEPTS DEVELOPED FOR ALGOL WILL SERVE AS A STARTING POINT FOR THE DISCUSSION OF OTHER PROGRAMMING LANGUAGES.

• DISCUSSION OF ALGOL IMPLEMENTATION WILL SERVE AS A STARTING POINT FOR A DISCUSSION OF MACHINE ORGANIZATION AND FOR THE BUILDING OF MODELS OF IMPLEMENTATION.
BASIC CONSTITUENTS OF A PROGRAMMING LANGUAGE

- **CONSTANTS** OF A NUMBER OF DIFFERENT TYPES SUCH AS INTEGERS, FLOATING POINT NUMBERS, LOGICAL CONSTANTS.

- **VARIABLES** (IDENTIFIERS) WHOSE VALUES MAY BE ELEMENTS OF A GIVEN CLASS OF CONSTANTS.

- **OPERATORS** — EACH OPERATOR HAS A DEGREE WHICH SPECIFIES THE NUMBER OF ARGUMENTS — THE TYPE PERMITTED FOR EACH ARGUMENT AND THE TYPE PERMITTED FOR THE RESULT MUST BE SPECIFIED.

- **EXPRESSIONS** — WHICH SPECIFY OPERATORS WITH THEIR ARGUMENTS AND YIELD A VALUE ON EVALUATION. AN EXPRESSION MAY HAVE SUBEXPRESSIONS WHOSE VALUES ARE ARGUMENTS OF HIGHER LEVEL EXPRESSIONS.

- **ASSIGNMENT STATEMENTS** WHOSE PRINCIPAL EFFECT IS TO CHANGE THE VALUE OF A VARIABLE.

- **BRANCHING STATEMENTS**, **CONDITIONAL STATEMENTS** AND **ITERATION STATEMENTS** WHICH DETERMINE THE FLOW OF CONTROL IN A PROGRAM.

- **DECLARATIONS** WHICH SPECIFY THE TYPE AND ATTRIBUTES OF VARIABLES.
CONSTITUENTS OF ALGOL

COMPLETE ALGOL PROGRAM — CONSISTS OF AN ALGOL BLOCK

BEGIN
  DECLARATIONS
  STATEMENTS
END

DATA DECLARATIONS
INTEGER X; X IS AN INTEGER
REAL Y, Z; Y AND Z ARE FLOATING POINT NUMBERS
BOOLEAN X; X IS A FLOATING POINT VARIABLE

ARRAYS OF DATA ELEMENTS
REAL ARRAY A[1:N]; A IS AN N-ELEMENT VECTOR OF FLOATING POINT NUMBERS

PROCEDURE DECLARATION
INTEGER PROCEDURE P(X,Y) SPECIFICATIONS BODY DECLARATION OF A TWO-PARAMETER PROCEDURE P WHICH PRODUCES A VALUE OF THE TYPE INTEGER. THE SPECIFICATIONS SPECIFY PARAMETER TYPES. THE BODY IS A PROGRAM WHICH SPECIFIES THE ACTION TO BE PERFORMED WHEN THE PROCEDURE IS CALLED.

LABEL AND SWITCH DECLARATIONS
LABEL L; (IMPLICIT DECLARATION)
SWITCH S: = L1; L2; L3; L4; S IS INITIALIZED TO A 4-ELEMENT ARRAY OF LABELS

STATEMENTS INCLUDE ASSIGNMENT STATEMENTS (X := X + 1;), BRANCHING STATEMENTS, CONDITIONAL STATEMENTS AND ITERATION STATEMENTS.

A BLOCK IS CONSIDERED TO BE A STATEMENT SO THAT STATEMENTS MAY HAVE BLOCKS NESTED INSIDE THEM.
CONSTANTS, VARIABLES AND EXPRESSIONS

CONSTANTS
CONSTANTS OF THE TYPE INTEGER 3; 4, 536
CONSTANTS OF THE TYPE REAL 3.5, 4, 372
CONSTANTS OF THE TYPE BOOLEAN TRUE, FALSE

OPERATORS WITH OPERANDS
INTEGER ADDITION 3 + 4
FLOATING POINT ADDITION 3.5 + 5, 3
COMPOSITION OF OPERATIONS 3 + 4 X 5
PRECEDENCE OF X OVER + (3 + 4) X 5
VARIABLES X + Y X Z
STATEMENTS Z: = X + Y;

TYPE SPECIFICATION
REAL X, Y, Z; INTEGER I, J;
Z: = X + Y;

MIXED EXPRESSIONS
Z: = X + I;
IMPLICIT CONVERSION FUNCTION
X + F CONVERT (I, REAL) FIRST CONVERT I TO REAL THEN USE FLOATING POINT ADDITION

RELATIONAL OPERATORS < ≤ = ≠ ≥ >
RELATION EXPRESSION, X > Y; NUMERICAL ARGUMENTS, BOOLEAN RESULT

BOOLEAN OPERATORS ¬∧∨⇒
BOOLEAN EXPRESSIONS; A ∧ B, BOOLEAN ARGUMENTS, BOOLEAN RESULTS
STRUCTURE OF ALGOL

STATEMENTS

V: = E;

LABELLED STATEMENT
L: x : = 1;
L: M: x : = 1;

MULTIPLE ASSIGNMENT
x: = y: = 1;

VALUE OF ASSIGNMENT STATEMENT IS VALUE OF ASSIGNED EXPRESSION

GO TO STATEMENT

GO TO L;
CONDITIONAL STATEMENTS AND CONDITIONAL EXPRESSIONS

STATEMENT

If B then S1 else S2
If x = 0 then y: = y + 1; else y: = y - 1;
If B then S
Equivalent to If B then S else (nothing)

EXPRESSION

If B then E1 else E2
y: = If x = 0 then y + 1 else y - 1;
y: = y + (If x = 0 then 1 else -1);

DESIGNATIONAL EXPRESSION

Go to If x = 0 then L1 else L2;
BLOCKS

COMPOUND STATEMENTS

BEGIN
  x := 5;
  y := 4
END

BLOCKS

BEGIN REAL K;
  K := X;
  X := Y;
  Y := K
END

K IS A LOCAL VARIABLE
IT IS NESTED ON ENTRY TO THE BLOCK, AND DESTROYED ON EXIT FROM THE BLOCK
Example: Nomenclature rules for nested blocks are as follows:

```
B: begin real x,y;
   x: = 3;
   y: = 4;
B1: begin real x,z;
    x: = 5;
    y: = 6;
    z: = 7;
    end;
print (x,y,z)
end
```

This sequence of ALGOL statements consists of a block B1 nested in a block B. The identifier y of the outer block can be used throughout the block B. However, the identifier x declared in the outer block cannot be used in the inner block because an identifier of the same name is declared in the inner block. The identifier x is bound in the inner block in the sense that if the two occurrences of the name x in the inner block were changed to another name, say u, then the computation defined by this program would be unaltered. The identifiers x, z of the inner block have meaning only in the inner block. In the print statement "print (x,y,z);" the identifiers x and y are associated with the declarations of x, y in the outer block and have the values x = 3, y = 6. The identifier z is undefined, so that this print statement would result in a diagnostic unless this program fragment were embedded in a block containing a declaration for the identifier z in its blockhead.
ITERATION STATEMENTS

Iteration statements have the following form:

\[ \text{for } V = \text{for list do } S \]

Execute the statement \( S \) for values of the variable \( V \) specified in the for list. It will be seen below that statements \( S \) may consist of arbitrarily complex nests of other statements, so that the restriction that the range of iteration be restricted to a single statement is not so severe as it appears.

The for-list elements may have one or more of the following three forms:

1. Individual expressions \( E \).

2. Expressions of the form "\( E_1 \) step \( E_2 \) until \( E_3 \)" indicating execution of \( S \) for values of \( V \) starting with \( E_1 \) and moving by increments of \( E_2 \) until \( E_3 \) is exceeded. Modification of \( E_2 \) and \( E_3 \) during execution of the statement \( S \) is permitted but not advised, since it may lead to trouble.

3. Expressions of the form "\( E \) while \( B \)", which specify execution of \( S \) with \( V = E \) as long as the value of \( B \) is true. In this case the statement \( S \) must be such that it can change the value of \( B \) to accomplish loop termination. \( S \) will normally also modify \( E \) when necessary.

The following example illustrates the use of a for statement to scan an \( N \)-element vector:

\[ \text{SUM: } = 0; \]
\[ \text{for } I = 1 \text{ step 1 until } N \text{ do } \]
\[ \text{SUM: } = \text{SUM} + A[I]; \]
FUNCTION AND STATEMENT TYPE PROCEDURES

procedure ADD (A, N, SUM);
  real array A; integer N; real SUM;
begin integer I;
  SUM: = 0;
  for I: = 1 step 1 until N do
    SUM: = SUM + A[I];
end

This declaration is a statement-type procedure. The first line specifies the name and formal parameters of the procedure. The second line specifies the types of formal parameters. The first two lines together are said to constitute the procedure heading. The remaining lines of the procedure constitute the procedure body, which in this case consists of a single block. The effect of the procedure is to SUM N elements of the array which constitutes the first parameter and store the result as the value of the third parameter.

Procedure Statement: ADD(X,15,S)

real procedure SUM(A,N);
  real array A; integer N;
begin integer I; real X;
  X: = 0;
  for I: = 1 step 1 until N do
    X: = X + A[I];
    SUM: = X;
end

This function-type procedure has one parameter less than the corresponding statement-type procedure, since the value is identified with the name and does not have to be explicitly specified by a parameter. The quantity X is used in the procedure body for accumulating the sum since an occurrence of SUM on the right-hand side of an assignment statement would be interpreted as a reentrant call of the procedure.

Call of Function Type Procedure

X: = SUM (A,15) + 2 x SUM(B,20);
PARAMETER CALLING

CALL BY VALUE - EVALUATE ON ENTRY TO PROCEDURE

CALL BY NAME - EVALUATE WHEN USED DURING PROCEDURE EXECUTION

```
REAL PROCEDURE P(A);
REAL A;
BEGIN
  K: = 5;
P: = A
END
```

IF A IS CALLED BY NAME, P(K) IS ALWAYS 5

IF A IS CALLED BY VALUE, P(K) IS GIVEN BY THE VALUE OF K ON ENTRY TO THE PROCEDURE.
THE STRUCTURE OF A COMPLETE PROGRAM CAN BE DESCRIBED IN TERMS OF THE STRUCTURE OF ITS FUNCTION MODULES.

ENTRY TO AND EXIT FROM FUNCTION MODULES IS IN LAST-IN-FIRST-OUT ORDER.

FUNCTION MODULES CAN BE STORED IN A STACK.
STATIC AND DYNAMIC NESTING

STATIC NESTING

DYNAMIC NESTING

\[
\begin{array}{c}
P \\
B1 \\
\end{array}
\begin{array}{c}
P2 \\
\end{array}
\]

\[\text{CALL P}\]

EXECUTION AT X

AT X, STATIC NESTING LEVEL IS 3, DYNAMIC NESTING LEVEL IS 4.

STATIC NESTING LEVEL IS A PROGRAM INVARIANT.

DYNAMIC NESTING LEVEL MAY BE ARBITRARILY DEEP WHEN CELLS ARE RECURSIVE.
REPRESENTATION OF IDENTIFIERS BY INTEGER PAIRS

(L, J) REPRESENTATION OF IDENTIFIERS

L - LEVEL OF STATIC NESTING

J - RELATIVE ADDRESS WITHIN ACTIVATION RECORD

(L, J) ADDRESS CAN BE USED FOR ACCESSING CURRENT ENVIRONMENT VECTOR MODEL

STATIC CHAIN MODEL

WITH STATIC CHAIN MODEL USE ADDRESS (R, J) WHERE R IS THE DIFFERENCE IN THE STATIC LEVEL OF NESTING BETWEEN THE POINT OF REFERENCE AND POINT OF USE OF THE IDENTIFIER.

R IS THE NUMBER OF STATIC CHAIN LINKS WHICH MUST BE FOLLOWED TO REACH THE ACTIVATION RECORD WHICH CONTAINS THE VALUE OF THE IDENTIFIER.
STRUCTURE OF ALGOL

RELATIVE ADDRESSING WITHIN PROCEDURE.
STORAGE FOR DECLARED QUANTITIES

BEGIN REAL x; REAL ARRAY A[1:10], B[m:n]; REAL y; ... END

VALUE OF x
- POINTER AND MAPPING INFORMATION FOR A
- POINTER AND MAPPING INFORMATION FOR B

VALUE OF y
- n-m+1 VALUES OF THE ELEMENTS B[m], ..., B[n]

ACTIVEATION-RECORD DATA STRUCTURE
CORRESPONDING TO THE BLOCKHEAD BEGIN
REAL x; REAL ARRAY A[1:10], B[m:n]; REAL y;..

DECLARED QUANTITIES
- INSTRUCTION POINTER
- STATIC CHAIN
- DYNAMIC CHAIN

STORAGE FOR ORGANIZATIONAL QUANTITIES
**STRUCTURE OF ALGOL**

**PROCEDURE ACTIVATION RECORDS**

Parameters called by value - store values

Parameters called by name - store procedure calls

Store value of function type procedures on completion.

<table>
<thead>
<tr>
<th>Parameters Called</th>
<th>List</th>
</tr>
</thead>
<tbody>
<tr>
<td>By Name</td>
<td></td>
</tr>
<tr>
<td>By Value</td>
<td></td>
</tr>
<tr>
<td>Instruction Pointer</td>
<td></td>
</tr>
<tr>
<td>Static Chain</td>
<td></td>
</tr>
<tr>
<td>Dynamic Chain</td>
<td></td>
</tr>
<tr>
<td>Function Value</td>
<td></td>
</tr>
</tbody>
</table>

17.14
ENVIRONMENT MODIFICATION

ON ENTRY TO AND EXIT FROM A BLOCK
ON ENTRY TO AND EXIT FROM A PROCEDURE
ON EVALUATION OF A PARAMETER CALLED BY NAME WITHIN A PROCEDURE
ON JUMP TO A LABEL
SYSTEM SYMBOLS – DENOTE FIXED INFORMATION STRUCTURES DEFINED BY THE SYSTEM.

BEGIN, FOR, +, 11.63

LOCAL IDENTIFIERS – LOCAL TO THE BLOCK CURRENTLY BEING EXECUTED.

NON LOCAL IDENTIFIERS – IN ENCLOSING BLOCKS

PROCEDURE PARAMETERS – ACCESS INFORMATION THROUGH POINT OF CALL.
  - BY VALUE
  - BY NAME
COMPILATION OF ALGOL PROGRAMS

EDIT FOR MORE CONVENIENT EXECUTION

EXPLICIT LABEL DECLARATIONS IN BLOCKHEADS

REPRESENT INTEGERS BY IDENTIFIER PAIRS

FUNCTION HEADING REPLACED BY STORAGE ALLOCATION INSTRUCTIONS

EXECUTABLE STRINGS ARE CONVERTED EITHER TO POSTFED NOTATION OR TO MACHINE LANGUAGE.
INFORMATION STRUCTURE MODEL FOR ALGOL

FIXED PROGRAM COMPONENT       P
STATEWORD COMPONENT            W
STACK COMPONENT                S
INPUT COMPONENT                IN
OUTPUT COMPONENT               OUT
OWN VARIABLE COMPONENT         X

I = (P, W, S, IN, OUT, X)

SPECIFY TRANSFORMATION F IN TERMS OF HOW THEY AFFECT INFORMATION COMPONENTS.

EMPHASIZE CREATION AND DELETION OF INFORMATION FIELDS.

CREATION OF ACTIVATION RECORDS ON ENTRY TO FUNCTION MODULES - DELETION ON EXIT FROM FUNCTION MODULES.

CREATION OF TEMPORARY INFORMATION FIELDS DURING EXPRESSION EVALUATION.

ASSIGNMENT STATEMENT MAY MODIFY AN INFORMATION FIELD IN THE INTERIOR OF THE STACK.
DATA MANAGEMENT

OUTLINE

1. THE STRUCTURE OF THE DATA MANAGEMENT ENVIRONMENT
2. THE JOB MANAGEMENT FUNCTION
3. THE EXTERNAL FILE SYSTEM
4. THE INTERNAL FILE SYSTEM
5. REVIEW OF DATA MANAGEMENT TECHNOLOGY
OBJECTIVES OF THE SESSIONS ON DATA MANAGEMENT

- TO PRESENT DATA MANAGEMENT CONCEPTS
- TO CONSTRUCT A FRAMEWORK FOR THE STUDY OF DATA MANAGEMENT PROBLEMS
- TO PROJECT AN APPROACH TO A MULTI-USER COMMON DATA BASE SYSTEM
- TO EXAMINE SOME CURRENT AND PROPOSED DESIGNS FOR DATA MANAGEMENT SYSTEMS
THE DATA MANAGEMENT ENVIRONMENT

THE DATA BASE

- THE ON-GOING DATA BASE
- THE PROBLEM OF SCALE
- SYSTEM RESPONSIBILITIES
  MULTI-LEVEL STORAGE MANAGEMENT
  ARCHIVING AND RECOVERY
  DATA INTEGRITY
THE DATA MANAGEMENT ENVIRONMENT

PROGRAM STRUCTURES AND THE DATA BASE

- THE PROGRAM DATA DECLARATION AS A TEMPLATE
- THE COMMON DATA BASE
- PROGRAM/DATA INDEPENDENCE
THE DATA MANAGEMENT SYSTEM

A DEFINITION

THE STORAGE, ASSOCIATION, AND RETRIEVAL OF

DIVERSE DATA ELEMENTS IN RESPONSE TO A VARIETY

OF PROCESSING DEMANDS

SOFTWARE TO

DEFINE DATA

USE IT

MAINTAIN IT

LINK IT TO PROGRAMS

LINK IT TO PEOPLE
THE DATA MANAGEMENT SYSTEM

- OBJECTIVES
  - CENTRAL RESPONSIBILITY FOR STORAGE, RETRIEVAL; AND REPORTING SERVICES TO THE USER.
  - CENTRAL RESPONSIBILITY FOR DATA INTEGRITY
  - SERVICES TO THE APPLICATION PROGRAMMER
  - REDUCTION OF PROGRAM DEVELOPMENT COSTS
  - INCREASE IN PROGRAM LIFE
  - ADAPTABILITY OF DATA STRUCTURES
  - OPTIMIZATION OF DATA UTILIZATION

- PRICE
  - "OVERHEAD"
  - SURRENDER OF TACTICAL DECISIONS
  - REDUCTION OF PROGRAMMER OPTIONS
PROGRAMMING COSTS
THE DATA MANAGEMENT ENVIRONMENT

ECONOMIC TRENDS

EDP EQUIPMENT COSTS

$ per Throughput Capacity

Time

Systems Design/Programming Costs

$ per Throughput Capacity

Time

Data Communication Costs

$ per Throughput Capacity

Time
APPLICATIONS

- BUSINESS DATA PROCESSING
- MANAGEMENT INFORMATION SYSTEMS
- COMMAND AND CONTROL
- INTERACTIVE SYSTEMS
- INFORMATION RETRIEVAL SYSTEMS
- MULTI–USER SYSTEMS
THE DATA MANAGEMENT ENVIRONMENT

ENVIRONMENT

- DATA CENTERS
- CENTRALIZED COMPUTATION SERVICES
- THE COMPUTING UTILITY
- THE OPERATIONS CONTROL CENTER
- THE CORPORATE DATA PROCESSING CENTER
THE DATA MANAGEMENT ENVIRONMENT

TYPICAL HARDWARE

- LARGE SCALE COMPUTER
- MASS RANDOM ACCESS STORES
- REMOTE ACCESS TERMINALS
COMPONENTS OF A FULL-SERVICE GENERALIZED DATA MANAGEMENT SYSTEM

- INTERNAL FILE SYSTEM
- EXTERNAL FILE SYSTEM
- JOB MANAGEMENT SYSTEM
- THE USERS
- SYSTEM SUPPORT FUNCTIONS
A FULL-SERVICE GENERALIZED DATA MANAGEMENT SYSTEM

- USERS
- ON LINE CONSOLE
- JOB MANAGEMENT
- INTERNAL FILE SYSTEM
- EXTERNAL FILE SYSTEM
THE DATA MANAGEMENT ENVIRONMENT

THE JOB MANAGEMENT SYSTEM

JOB MANAGEMENT SERVICES

JOB TASKS
USER COMMANDS

- CONTROL
- FILE MANIPULATION
- RELATION MANIPULATION
- FIELD MANIPULATION
- BLOCK TRANSFER
EXAMPLE OF RESPONSE TO *DEFINE FILE.

*DEFINE FILE
FILE DEFINITION, DO YOU WANT INSTRUCTIONS: YES PROVIDE THE FOLLOWING (12 CHARACTER MAXIMUM FOR EACH):
NAME OF FILE
TYPE OF FILE (NAMED OR NUMBERED)
THE NAME OF EACH DATA FIELD FOLLOWED BY ITS CODING ACCEPTABLE CODINGS ARE THE FOLLOWING:
BCD, INTEGER, FLT. POINT, BCD LIST, INTEGER LIST, FLT. PT. LIST

A CARRIAGE RETURN MUST FOLLOW EACH INPUT TERM.
THE WORD * DONE TERMINATES INPUT.

COMPUTER
NAMED
RENTAL
INTEGER
ADD TIME
FLT. POINT
CYCLE TIME
FLT. POINT
CORE STORAGE
INTEGER
DRUM STORE
INTEGER
WORD SIZE
BCD
SPEC FEATURE
BCD LIST
*DONE
EXAMPLE (CONTD)

THE INPUT TABLE FOLLOWS:

<table>
<thead>
<tr>
<th>COMPUTER NAMED</th>
<th>RENTAL INTEGER</th>
<th>ADD TIME FLT. POINT</th>
<th>CYCLE TIME FLT. POINT</th>
<th>CORE STORAGE INTEGER</th>
<th>DRUM STORAGE INTEGER</th>
<th>WORD SIZE BCD</th>
<th>SPEC FEATURE BCD LIST</th>
</tr>
</thead>
</table>

IS THIS WHAT YOU WANT. IF NOT, TYPE "NO" AND START AGAIN.
YES
FILE SET-UP COMPLETED.

COMMAND EXECUTED.

GIVE COMMAND OR TYPE *CHOICES.
EXAMPLE OF RESPONSE TO *INPUT ENTRIES.

*INPUT ENTRIES

TYPE:
FILE NAME
*INSTRUCTIONS OR *NO

COMPUTER
*INSTRUCTIONS

FOR EACH ENTRY TO BE ADDED:

1. WAIT UNTIL "READY" IS TYPED
2. LIST CONTENTS OF THE DATA FIELDS
   A. IF SOME FIELD IS ITSELF A LIST,
      A BLANK LINE SIGNIFICATES THE END OF THE LIST
   B. FORMATS ARE:
      FOR BCD: FIELD LENGTH=6, LEFT JUSTIFY DATA
      FOR INTEGERS: FIELD LENGTH=12, RIGHT JUSTIFY DATA
      FOR FLT. PT.: FIELD LENGTH=16; PROVIDE DECIMAL PT.
3. TYPE THE PARENT OF THIS ENTRY FOR EACH RELATION LISTED
4. TO TERMINATE INPUT OF ENTRIES, PRESS CR AFTER "READY" IS TYPED
EXAMPLE (CONTD.)

<table>
<thead>
<tr>
<th>DATA FIELDS</th>
<th>CODING</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>BCD</td>
</tr>
<tr>
<td>ADD TIME</td>
<td>FLOATING POINT</td>
</tr>
<tr>
<td>CORE STORAGE</td>
<td>INTEGER</td>
</tr>
<tr>
<td>CYCLE TIME</td>
<td>FLOATING POINT</td>
</tr>
<tr>
<td>DRUM STORE</td>
<td>INTEGER</td>
</tr>
<tr>
<td>RENTAL</td>
<td>INTEGER</td>
</tr>
<tr>
<td>SPEC FEATURE</td>
<td>BCD LIST</td>
</tr>
<tr>
<td>WORD SIZE</td>
<td>BCD</td>
</tr>
</tbody>
</table>

RELATIONS
THERE ARE NO RELATIONS

READY

IBM 7094 11
1.4
32
1.4
186
160
IN'RUP
16XR'S
FLT. PT
IN ADD
648

18.16.1
EXAMPLE OF RESPONSE TO *SEARCH FILE.

*SEARCH FILE

THE ACTIVE FILES ARE:

COMPUTER
HOME ADDRESS
STREET

PROVIDE FILE NAME: COMPUTER

(FILE DESCRIPTION)

COMPUTER IS A FILE WITH NAMED ENTRIES.
NO. OF DATA FIELDS PER ENTRY = 7

SAMPLE ENTRY FOLLOWS:

ENTRY: CDC 3600
ADD TIME : 2.00
CORE STORAGE : 262
CYCLE TIME : 1.50
DRUM STORE : 0
RENTAL : 55
SPEC FEATURE : IN'RUP
6XR'S
FLT. PT
IN'ADD
WORD SIZE : 488
EXAMPLE (CONTD)

(START OF SEARCH)

PROVIDE FIELD NAME: CYCLE TIME
PROVIDE CONDITION (EQ,LT,GT,LTOREQ,GTOREQ): LT
PROVIDE TEST VALUE (FLTG. POINT NUMBER): 4.0
DO YOU WNT FULL ENTRIES PRINTED: YES

(START OF SUBFILE)

ENTRY: CDC 3600
ADD TIME : 2.00
CORE STORAGE : 262
CYCLE TIME : 1.50
DRUM STORE : 0
RENTAL : 55
SPEC FEATURE : IN'RUP
  6XR'S
  FLT, PT
  IN'ADD
WORD SIZE : 488
FILE MANIPULATION COMMANDS

- DEFINE FILE
- INPUT ENTRIES
- SEARCH FILE
- LIST FILES
- PRINT FILE
- FIND VALUE
- DELETE FILE
EXAMPLE (CONTD)

MANUAL MODE

LIST THE NAMES OF THE PARENT ENTRIES FOLLOWED BY THE NAMES OF THEIR RELATED SUBFILE ENTRIES. TO TERMINATE THE LIST OF SUBFILE ENTRIES LEAVE A LINE BLANK. TO TERMINATE INPUT LEAVE ANOTHER LINE BLANK. WAIT FOR THE WORK "READY" BEFORE TYPING IN EACH GROUP OF PARENT AND LINKEES.

WHICH MODE DO YOU WANT* MANUAL

READY

WOBURN
ALLEN MARGAR
ATHANS MICHAEL
CORR DAVID F

READY

CAMBRIDGE
ANDERSON ALL
COHEN MITCHE
CURTISS ARTHUR
FALB PETER L
THE USERS OF THE DATA MANAGEMENT SYSTEM

MANAGERS
- Ad Hoc Reports
- Queries
- Administration

PROGRAMMERS
- Logical Data Services
- Data-Independent Programs

ANALYSTS
- Build System Jobs
- General-Purpose Modularity

DATA ADMINISTRATOR
- Data Base Control
- Monitor Use
- Control Access
RELATION MANIPULATION COMMANDS

- DEFINE RELATION
- SEARCH RELATION
- LIST RELATIONS
- DESCRIBE RELATIONS
- FIND PARENT
- FIND LINKEE
- RELATE ENTRY
- DELETE RELATION
FIELD MANIPULATION COMMANDS

- DEFINE DATA FIELD
- DELETE DATA FIELD
- DEFINE FIELD VALUE
BLOCK DATA TRANSFER COMMANDS

- READ CARDS
- WRITE TAPE
CELL STRUCTURE OF SAMPLE FILE

A CELLS

A1

A 4 | A 30
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DAT
BASIC FILES

FILE NAME

ACTIVE FILE

STARTING ENTRY
SIZE OF ENTRY
TYPE
EMPTY SPACE

FIELD NAME

DATA FIELD FILE

POSITION IN ENTRY
CODING

RELATION NAME

RELATIONS FILE

RELATION TYPE
ORDERING RULE
ORDER FIELD
RELATION LINKS
LISTAR

LINK TYPES

LINK FIELD

KEY

ADDRESS

KEY MEANING OF ADDRESS

P POINTER TO FILE ENTRY

B BRANCH TO SUBFILE

D DESCEND TO NEXT FILE ENTRY

A ASCEND TO PRECEDING ENTRY

R RETURN FROM SUBFILE TO PARENT FILE

U UNUSED LINK FIELD

C CELL LINK

E EMPTY SUBLIST INDICATOR

ASSOCIATIVE LINKS

18. 25
ONE-WAY LIST

TWO-WAY LIST
ONE-WAY RING

TWO-WAY RING

18, 27
STATEMENT GROUPING

DO;
X = 5;
Y = 3;
END;

DO STATEMENT

END STATEMENT

DO I = 1 BY 1 TO N;
SUM = SUM A[I];
END;

DO I = E1 BY E2 TO E3; E1, E2, E3 ARE INITIALIZED
BY VALUE
FOR I = E1 STEP E2 UNTIL E3 DO S;
E1, E2, E3 ARE INITIALIZED BY NAME
STRUCTURE OF PLI

BLOCKS

BEGIN

STATEMENTS
AND DECLARATIONS

END

DECLARATIONS NEED NOT OCCUR AT THE BEGINNING OF THE BLOCK BUT ARE ASSUMED EXECUTED AS THOUGH THEY WERE AT THE BEGINNING OF THE BLOCK.

SIMULTANEOUS DECLARATIONS

LAYERS OF DECLARATION AS IN CPL

DYNAMIC DECLARATIONS - NEW DECLARATION EVERY TIME IT IS ENCOUNTERED DURING EXECUTION - LIKE A PROCEDURE CALL WHOSE EFFECT IS TO DECLARE RATHER THAN TO EXECUTE.
PROCEDURES

NAME: PROCEDURE(P) SPECIFICATIONS

DECLARATIONS AND STATEMENTS

END;

NAME IS LIKE A LABEL

PARAMETERS ARE CALLED BYREFERENCE

RETURN STATEMENT

RETURN (EXP) VALUE OF EXP IS RETAINED TO POINT OF CALL
DECLARATIONS

DECLARE NAME ATTRIBUTES
DECLARE (N1,N2) A
DECLARE (N1 A1, N2 A2) A3

CLASSIFICATION OF ATTRIBUTES

TYPE ATTRIBUTES - LIKE DATA TYPES OF ALGOL - SPECIFY THE RANGE OF VALUES AND SET OF OPERATIONS APPLICABLE TO THE IDENTIFIER.

STRUCTURE ATTRIBUTES - SPECIFY SUBSTRUCTURE OF THE INFORMATION STRUCTURE DENOTED BY THE IDENTIFIER.

SCOPE ATTRIBUTES - SPECIFY THE RANGE OF STATEMENTS OF THE STATIC SOURCE PROGRAM OVER WHICH THE IDENTIFIER HAS MEANING.

STORAGE ATTRIBUTES - SPECIFY THE LIFETIME OF THE INFORMATION STRUCTURE.

19.4
DATA ATTRIBUTES

BASE ATTRIBUTES - DECIMAL, BINARY

SCALE ATTRIBUTES - FIXED, FLOAT

MODE ATTRIBUTES - REAL, COMPLEX

PRECISION ATTRIBUTES - (N, M)

DECLARE A DECIMAL FIXED REAL (3, 2);

DEFAULT ATTRIBUTES

BINARY FIXED REAL

DEFAULT PRECISION IS IMPLEMENTATION-DEFINED
CHARACTERS, LOGICALS AND POINTERS

NON-ARITHMETIC DATA TYPES

CHARACTERS AND CHARACTER STRINGS

DECLARE A

BITS AND BIT-STRINGS

STRING CONSTANTS 'ABC', '0100'B

STRING VARIABLES X 'ABC'; Y '0100'B

POINTERS AND POINTER VALUED VARIABLES

POINTER P, Q;

FUNCTION ADDR(X) - RETURNS POINTER TO X

P = ADDR(A)
P \rightarrow A = 5
ARRAYS AND STRUCTURES

VARIABLE DIMENSIONS - LOWER AND UPPER BOUNDS
DECLARE A(1, 5:10);

ARRAYS ARE RESTRICTED TO BE RECTANGULAR, AND TO HAVE ALL
ELEMENTS BE OF THE SAME TYPE

STRUCTURES - DATA ELEMENTS MAY BE OF DIFFERENT TYPES -
NOT RESTRICTED TO BE RECTANGULAR

DATA ELEMENTS OF A STRUCTURE ARE TERMINAL VERTICES OF A TREE

LEVEL NUMBERS

DECLARE
1 PAYROLL
2 NAME
2 HOURS
3 REGULAR
3 OVERTIME
2 RATE

DECLARE
1 PAYROLL
2 NAME
2 HOURS
3 REGULAR
3 OVERTIME
2 RATE
ATTRIBUTE AND NAMES OF STRUCTURE COMPONENTS

ASSOCIATE ATTRIBUTES WITH DATA ITEMS

DECLARE
  1 PAYROLL,
  2 NAME CHARACTER (50) VARYING,
  2 HOURS,
  3 REGULAR FIXED,
  3 OVERTIME FIXED,
  2 RATE FLOAT;

TREE NAMES

  PAYROLL, HOURS, REGULAR

DEFAULT NAMES IF UNAMBIGUOUS

  PAYROLL, REGULAR
  HOURS, REGULAR
  REGULAR

19.8
STRUCTURE OF PL/I

SCOPE ATTRIBUTES

SCOPE - INTERNAL OR EXTERNAL

INTERNAL - KNOWN ONLY WITHIN THE BLOCK IT IS DECLARED

EXTERNAL - GLOBALLY KNOWN

EXTERNAL PROCEDURE NAME - LIKE FORTRAN SUBROUTINE NAME

A PL/I PROGRAM CONSISTS OF A GROUP OF EXTERNAL PROCEDURES

EXTERNAL DATA NAME - LIKE COMMON IN FORTRAN
STORAGE ALLOCATION ATTRIBUTES

PL/I HAS FORTRAN, ALGOL AND LIST PROCESSING MODES OF STORAGE ALLOCATION.

FORTRAN MODE - STATIC
LIFETIME OF STATIC STRUCTURES IS THE WHOLE COMPUTATION

ALGOL MODE - AUTOMATIC
LIFETIME OF AUTOMATIC STRUCTURES IS THE BLOCK ON WHICH THEY ARE DECLARED.

LIST PROCESSING MODE - CONTROLLED
A CONTROLLED STORAGE ALLOCATION DECLARATION CREATES A TEMPLATE FOR THE DECLARED STRUCTURE.

INSTANCES OF A STRUCTURE CREATED BY A CONTROLLED STORAGE ALLOCATION DECLARATION ARE CREATED BY ON ALLOCATE COMMAND AND DELETED BY A FREE COMMAND.
CONTROLLED STORAGE ALLOCATION

DECLARE 1 A CONTROLLED
2 X FIXED
2 Y POINTER

<table>
<thead>
<tr>
<th>X:</th>
<th>FIXED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y:</td>
<td>POINTER</td>
</tr>
</tbody>
</table>

ALLOCATE A
A·X = 5
X = X + 1
A·Y = ADDR(Z)
ALLOCATE A
FREE A
FREE A

MULTIPLE ALLOCATION CAUSES AUTOMATIC STACKING OF INSTANCES.

ACCESS TO INSTANCES THROUGH POINTERS

ALLOCATE A SET P
ALLOCATE A SET Q
P → A·X = 5
Q → X = 6
DECLARE 1 A BASED (P)
2 X FIXED
2 Y POINTER
ALLOCATE A
ALLOCATE A

WHEN SECOND COPY IS NESTED, FIRST COPY IS DESTROYED.
CREATE THE FOLLOWING THREE-ELEMENT LIST.

```
<table>
<thead>
<tr>
<th>HEAD</th>
<th>A: EMPTY</th>
<th>B: POINTER</th>
</tr>
</thead>
</table>
```

```
<table>
<thead>
<tr>
<th>A: EMPTY</th>
<th>B: POINTER</th>
</tr>
</thead>
</table>
```

```
<table>
<thead>
<tr>
<th>A: EMPTY</th>
<th>B: NULL</th>
</tr>
</thead>
</table>
```

DECLARE (Q, HEAD) POINTER;
DECLARE 1 ELEMENT BASED (P),
2 A FIXED,
2 B POINTER;
ALLOCATE ELEMENT;
HEAD = P;
Q = P;
ALLOCATE ELEMENT;
Q → B = P;
Q = P;
ALLOCATE ELEMENT;
Q → B = P;
B = NULL;

This program declares Q and HEAD to be of type POINTER and ELEMENT to be a structure based on P. The instruction "ALLOCATE ELEMENT" automatically sets P to the most recent instance of ELEMENT. The assignment statements "HEAD = P; Q = P;" assign the value of the pointer P to the pointers Q, HEAD.

When the second instance of ELEMENT has been created, Q points to the first instance, and "Q → B = P" sets the pointer B of the first instance to point to the second instance. Similarly after creation of the third instance of ELEMENT, "Q → B = P" sets the pointer B in the second instance to point to the third instance. Finally "B = NULL", which is equivalent to "P → B = NULL" sets the current instance of B to the special pointer value NULL, which indicates the end of the list.
STRUCTURE DECLARATION FOR LIST ELEMENT

DECLARE 1 LISPCELL BASED(P),
2 CAR POINTER,
2 CDR POINTER,
2 MODE BIT(6);

This declaration specifies the basic format of a list cell in LISP to consist of two pointer fields named CAR and CDR and a 6-bit mode field.

HEAD AND TAIL OPERATIONS

HEAD: PROCEDURE(P) POINTER;
DECLARE 1 ELEMENT BASED(P),
2 CAR POINTER,
2 CDR POINTER;
RETURN(CAR);
TAIL: ENTRY(P);
RETURN(CDR);
END HEAD;

This pointer-valued procedure has two entry points, HEAD and TAIL. The declaration of ELEMENT specifies the structure pointed to by the procedure parameter P. The structure itself is assumed to have been created outside the procedure and to be an element of a list of structures of the kind arising in LISP. The call HEAD(P) returns with a value given by the pointer in the first field of the structure pointed to by P while the call TAIL(P) returns with a value given by the second field of the structure pointed to be P.

CONS OPERATOR

CONS: PROCEDURE(P, Q) POINTER;
DECLARE 1 ELEMENT BASED(X),
2 LEFT POINTER,
2 RIGHT POINTER;
ALLOCATE ELEMENT;
LEFT = P;
RIGHT = Q;
RETURN(X);
END CONS;

This pointer-valued procedure has two pointer-valued parameters, P and Q. It allocates an instance of the structure ELEMENT, stores the pointers P and Q in the first and second registers of the newly created structure, and delivers a pointer to the newly created structure as its value.
FEATURES WHICH FACILITATE LIST PROCESSING

VARIABLES OF TYPE POINTER WHICH ALLOW LINKS BETWEEN INFORMATION STRUCTURES TO BE EXPLICITLY SPECIFIED AND MANIPULATED

STRUCTURE DECLARATIONS WHICH ALLOW LIST ELEMENTS CONTAINING SEVERAL POINTER AND VALUE FIELDS OF DIFFERENT TYPES TO BE EXPLICITLY DECLARED

CONTROLLED STORAGE ALLOCATION, WHICH allows structures to be dynamically created and deleted as they are required.

IN A GIVEN LIST PROCESSING LANGUAGES ALL LIST STRUCTURES ARE FORMED OUT OF LIST ELEMENTS OF A LIMITED NUMBER OF PRIMITIVE TYPES

IN PL/I NEW PRIMITIVE TYPES OF LIST ELEMENTS MAY BE DEFINED BY STRUCTURE DECLARATIONS
IMPLEMENTATION OF CONTROLLED STORAGE ALLOCATION

CREATION AND DELETION IN UNPREDICTABLE ORDER

INSTANCES CANNOT BE STORED IN A STACK

FREE STORAGE AREA IS REQUIRED

ALLOCATE AND RETURN BLOCKS AS REQUIRED

FRAGMENTATION OF MEMORY

GARBAGE COLLECTION IS SOMETIMES NECESSARY
STATEMENT GROUPING IN FORTRAN, ALGOL AND PL/I

PURPOSES OF STATEMENT GROUPING

1. TO DELIMIT A PROCEDURE WHICH MAY BE CALLED IN SEVERAL PLACES
2. TO DELIMIT THE SCOPE OF NAMES
3. TO GROUP STATEMENTS FOR CONTROL PURPOSES
4. TO SPECIFY THE LIFETIME OF INFORMATION ITEMS

<table>
<thead>
<tr>
<th>Purpose</th>
<th>FORTRAN</th>
<th>ALGOL</th>
<th>PL/I</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Delimit procedures</td>
<td>Program unit</td>
<td>begin-end (procedure heading)</td>
<td>PROCEDURE-END</td>
</tr>
<tr>
<td>2. Scope of nomenclature</td>
<td>Program unit</td>
<td>begin-end</td>
<td>PROCEDURE-END (INTERNAL EXTERNAL)</td>
</tr>
<tr>
<td>3. Unit for control purposes</td>
<td>DO-loop</td>
<td>begin-end (for clause)</td>
<td>BEGIN-END DO-END (DO-statement)</td>
</tr>
<tr>
<td>4. Lifetime of information</td>
<td>not needed</td>
<td>begin-end (own)</td>
<td>BEGIN-END for AUTOMATIC (STATIC AUTOMATIC CONTROLLED)</td>
</tr>
</tbody>
</table>
STRUCTURE OF PL/I

INTERRUPT FUNCTION MODULES

CONDITION PREFIXES

(ZERODIVIDE): L: X = A/B;

ON STATEMENT

ON CONDITION ACTION

LIKE A PROCEDURE DECLARATION

ENTRY WHEN (INTERRUPT) CONDITION OCCURS RATHER THAN BY EXPLICIT CALL - INTERRUPT FUNCTION MODULE

BEGIN BLOCKS - ENTRY AND EXIT IN LINE

PROCEDURE BLOCKS - CALL AND RETURN

ON MODULES - INTERRUPT AND RETURN

ENTRY AND EXIT FOR ALL THREE TYPES IS MUTUALLY IN A LAST IN FIRST OUT ORDER

ACTIVATION RECORDS MAY BE STORED IN A STACK
CHARACTERISTICS OF THE SYSTEM

- MULTI-USE
- ON-GOING DATA BASE
- COMMON DATA BASE
- JOB LIBRARY
- PREREQUISITE SCHEDULING
- REAL-TIME SCHEDULING
THE COHERENT SYSTEM CONCEPT

- COHERENCE OF PROGRAMS
- COHERENCE OF DATA
- COHERENCE OF CONTROL
- RESPONSIBILITY FOR COHERENCE:

  PROGRAMMER
  PROGRAM TRANSLATORS
  SYSTEM
THE JOB MANAGEMENT FUNCTION

DATA TRANSFORMATION FUNCTIONS

BY PROGRAM

\[ \text{BASIS} \rightarrow \text{OPERATOR} \rightarrow \text{RESULT} \]

BY TABLE

\[ [\text{BASIS, RESULT}] \rightarrow \text{OPERATOR} \rightarrow (\text{BASIS}) = \text{RESULT} \]

FUNCTIONAL NOTATION

\[ \text{OPERATOR} (\text{BASIS}) = \text{RESULT} \]

\[ F (X) = Y \]

\[ \text{SQRT} (4) = 2 \]
THE JOB MANAGEMENT FUNCTION

INTERFACE BETWEEN TASK, JOB MANAGEMENT SYSTEM, AND FILE SYSTEM

- JOB REQUEST
- JOB DEFINITION
- PROGRAM DEFINITION
- VIRTUAL STORE
- DATA BASE
- TASK"
**JOB DEFINITION EXAMPLE**

**FILE NAME** ---**EXTRACT** ---**SORT** ---**PRINT**

**FIELDS** ---**KEY** ---**ITEM NAME** ---**SAVE** ---**DATA BASE**

**DEFINE JOB**: PERSONNEL LIST (CONDITION, ITEM NAME)

- **EXTRACT** (PERSONNEL FILE, (NAME, EMPL. NO, POS), CONDITION) = *1
- **SORT** (*1, NAME) = *2
- **SAVE** (*2, ITEM NAME)
- **PRINT** (*2).

20.3.1
THE JOB MANAGEMENT FUNCTION

USER INTERFACE AND SYSTEM LANGUAGES

USER

PROGRAMMER

COMMAND (JOB REQUESTS)

PROGRAM ENTRY

DATA ADMINISTRATOR

DATA/DIRECTORY CHANGE

JOB DEFINE/RUN

COHERENT SYSTEM

PROGRAM LIBRARY

PROCEDURAL

DIRECTORY

DESCRIPTIVE

DATA BASE

DECLARATIVE

20.5
USER LANGUAGES

- JOB REQUEST (COMMAND) LANGUAGE
- DATA ITEM DEFINITION LANGUAGE
- DATA ITEM INPUT LANGUAGE
- JOB DESCRIPTION LANGUAGE
- DATA SERVICE REQUEST LANGUAGE
- ON-LINE (INTERPRETIVE) COMPUTATIONAL LANGUAGES
- COMPILER LANGUAGE
- MACRO ASSEMBLER LANGUAGE
THE JOB MANAGEMENT FUNCTION

THE NEED FOR LANGUAGE ADAPTABILITY

- CHANGES IN CAPABILITY
- CHANGES IN USERS
- HUMAN FACTORS EXPERIMENTS
THE JOB MANAGEMENT FUNCTION

SYNTAX DIRECTED PROCESSING

- SYNTAX
- SEMANTICS
- SCANNER/ANALYZER
INSCAN: TRANSLATION MODE

THE JOB MANAGEMENT FUNCTION
INSCAN: INTERPRETIVE MODE
GENERAL INSCAN CONFIGURATION

ACTION GRAPH

INPUT STREAM

INSCAN

OUTPUT STREAM

INPUT POINTER

OUTPUT POINTER

ACTORS

WORKING STORAGE
PHASES OF USER LANGUAGE PROCESSING

(A) "ASSEMBLY" TIME

STAG TRANSLATOR AGT

STRING ACTION GRAPH (STAG)

INSCAN

ABSOLUTE ACTION GRAPH TABLE (AGT)

(B) "EXECUTION" TIME

USER–LANGUAGE AGT

USER–LANGUAGE INPUT STRING

INSCAN

USER–LANGUAGE OUTPUT STRING
## THE JOB MANAGEMENT FUNCTION

### EXAMPLES OF SYNTAX SPECIFICATIONS

<table>
<thead>
<tr>
<th>JM</th>
<th>PARENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>[\langle JM \rangle ::= \text{JOHN MARSHA}</td>
<td>\text{JOHN} \langle JM \rangle \text{MARSHA}]</td>
</tr>
</tbody>
</table>

(A) BNF

(B) TRANSITION DIAGRAM

(C) SYNTAX CHART

(D) ACTION GRAPH
THE JOB MANAGEMENT FUNCTION

ACTION GRAPH SYMBOLS

- GRAPH NAME: $\tau$ IS DEFINED BY PATH $\delta$
- SCAN: READ INPUT SYMBOL AND MATCH $\alpha$
- CHOICE: TRY ALTERNATIVES 1 AND 2
- SUBGRAPH: EXECUTE GRAPH $\alpha$ AND RETURN
- RECURSE: EXECUTE THIS GRAPH RECURSIVELY AND RETURN
- EXTERNAL ACTION: DO SUBROUTINE $\pi$ AND RETURN
- INTERNAL ACTION: DO OPERATIONS $\omega$
- END: RETURN TO PARENT GRAPH
SAMPLE INPUT STRING FOR THE INFIX RECOGNIZER

\[(A + B) - (A - B)\]

1 2 3 4 5 6 7 8 9 10 11
OPERATION OF THE INFIX RECOGNIZER

<table>
<thead>
<tr>
<th>PORTION OF STRING RECOGNIZED</th>
<th>RECOGNIZED BY</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>TERM</td>
</tr>
<tr>
<td>B</td>
<td>TERM</td>
</tr>
<tr>
<td>A + B</td>
<td>NEST</td>
</tr>
<tr>
<td>(A + B)</td>
<td>TERM</td>
</tr>
<tr>
<td>A</td>
<td>TERM</td>
</tr>
<tr>
<td>B</td>
<td>TERM</td>
</tr>
<tr>
<td>A - B</td>
<td>NEST</td>
</tr>
<tr>
<td>(A - B)</td>
<td>TERM</td>
</tr>
<tr>
<td>(A + B) - (A - B)</td>
<td>NEST</td>
</tr>
</tbody>
</table>
THE JOB MANAGEMENT FUNCTION

INFIX-TO-SUFFIX TRANSLATOR

NEST

TERM

TERM

WRITE "+"

WRITE "-"

TERM

TERM

SAVE INPUT POINTER

A

B

NEST

COPY

INPUT POINTER

(  

2  

3  

5  

9  

11+11

9

## OPERATION OF THE INFIX-TO-SUFFIX TRANSLATOR

<table>
<thead>
<tr>
<th>PORTION OF STRING RECOGNIZED</th>
<th>RECOGNIZED BY</th>
<th>SYMBOL ADDED TO OUTPUT STRING</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>TERM</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>TERM</td>
<td>B</td>
</tr>
<tr>
<td>A + B</td>
<td>NEST</td>
<td>+</td>
</tr>
<tr>
<td>(A + B)</td>
<td>TERM</td>
<td>A</td>
</tr>
<tr>
<td>A - B</td>
<td>NEST</td>
<td>-</td>
</tr>
<tr>
<td>(A - B)</td>
<td>TERM</td>
<td></td>
</tr>
<tr>
<td>(A + B) - (A - B)</td>
<td>NEST</td>
<td>-</td>
</tr>
</tbody>
</table>
THE JOB MANAGEMENT FUNCTION

STAG SYNTAX

ACTION GRAPH

NAME : CLAUSE

CLAUSE

TAG : EXECUTE NAME

EXECUTE NAME

RECURSE

LITERAL

CHOICE

GOOD

GOTO

CALL

WRITE

COPY

SAVE INPUT POINTER
STAG - LANGUAGE ACTION GRAPHS FOR THE INFIX-TO-SUFFIX TRANSLATOR

NEST: EXECUTE TERM; CHOICE (3, 5, 9);

3: "+"; EXECUTE TERM; WRITE "+"; GOOD;

5: "−"; EXECUTE TERM; WRITE "−";

9: GOOD.

TERM: SAVE INPUT POINTER; CHOICE (2, 3, 5);

2: "A"; 9: COPY; GOOD;

3: "B"; GOTO 9;

5: "("; EXECUTE NEST; ")"; GOOD.