PROCESS MANAGEMENT IN MPE-III

CHRIS MOELLER
HP CUPERTINO.

- CONTENTS -

USER VIEW OF PROCESS

- AN ENTITY IN MPE THAT WILL ALLOW A USER'S CODE TO BE EXECUTED.

- SOMETHING THAT REQUIRES PH CAPABILITY TO USE AND HAS SOME NICE INTRINSICS ASSOCIATED WITH IT.

  - o CREATE, KILL, TERMINATE
  - o ACTIVATE, SUSPEND
  - o GETPROCID, GETPROCINFO, GETORIGIN, FATHER
  - o MAIL, SENDMAIL, RECEIVEMAIL
  - o GETPRIORITY

PROCESS DEFINITION

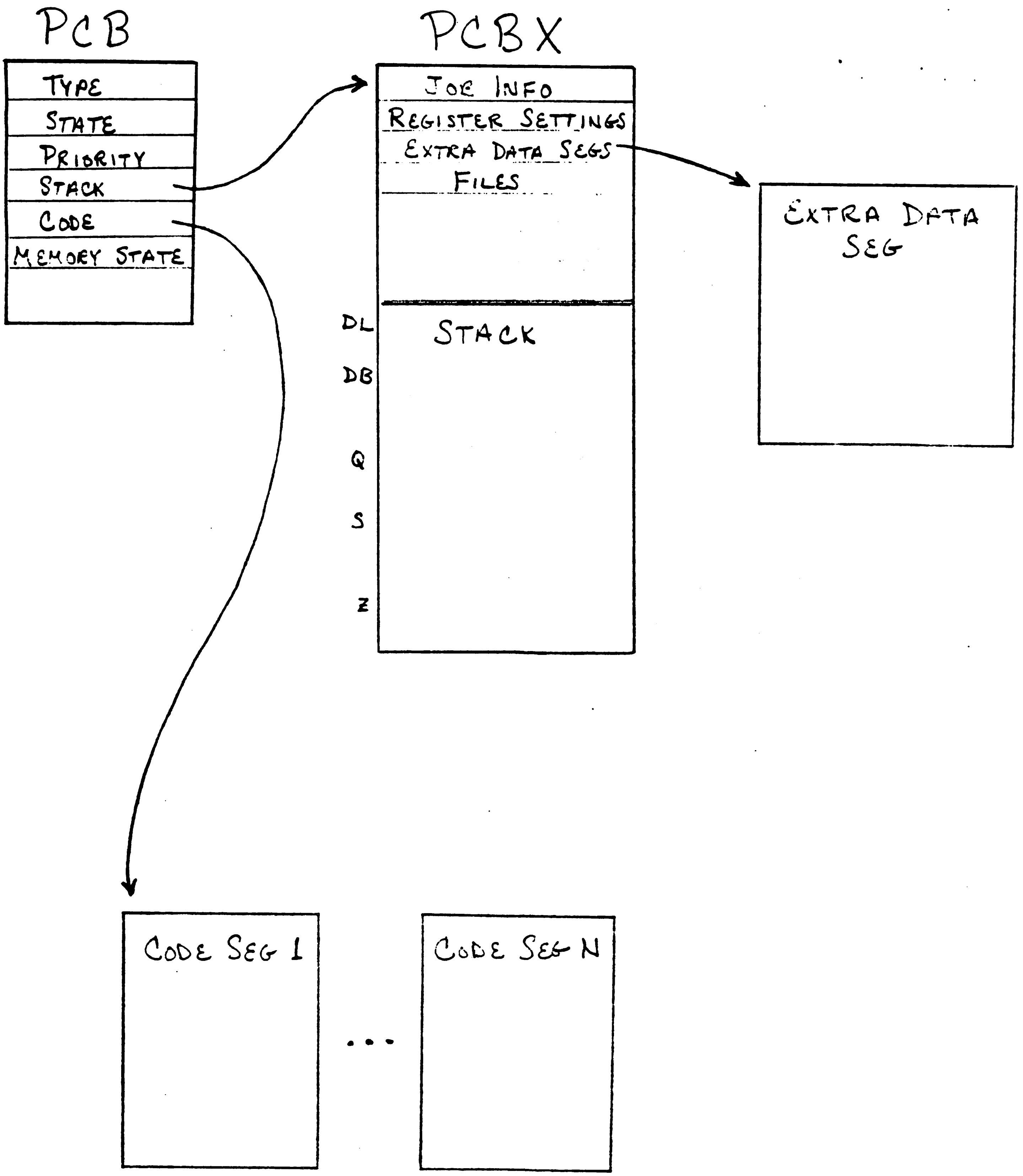
A PROCESS IS THE ENTITY IN MPE WHICH CAN ACCOMPLISH WORK. IT IS THE ENTITY WHICH ALLOWS SYSTEM RESOURCES TO BE SHARED.

A PROCESS MUST ALWAYS HAVE

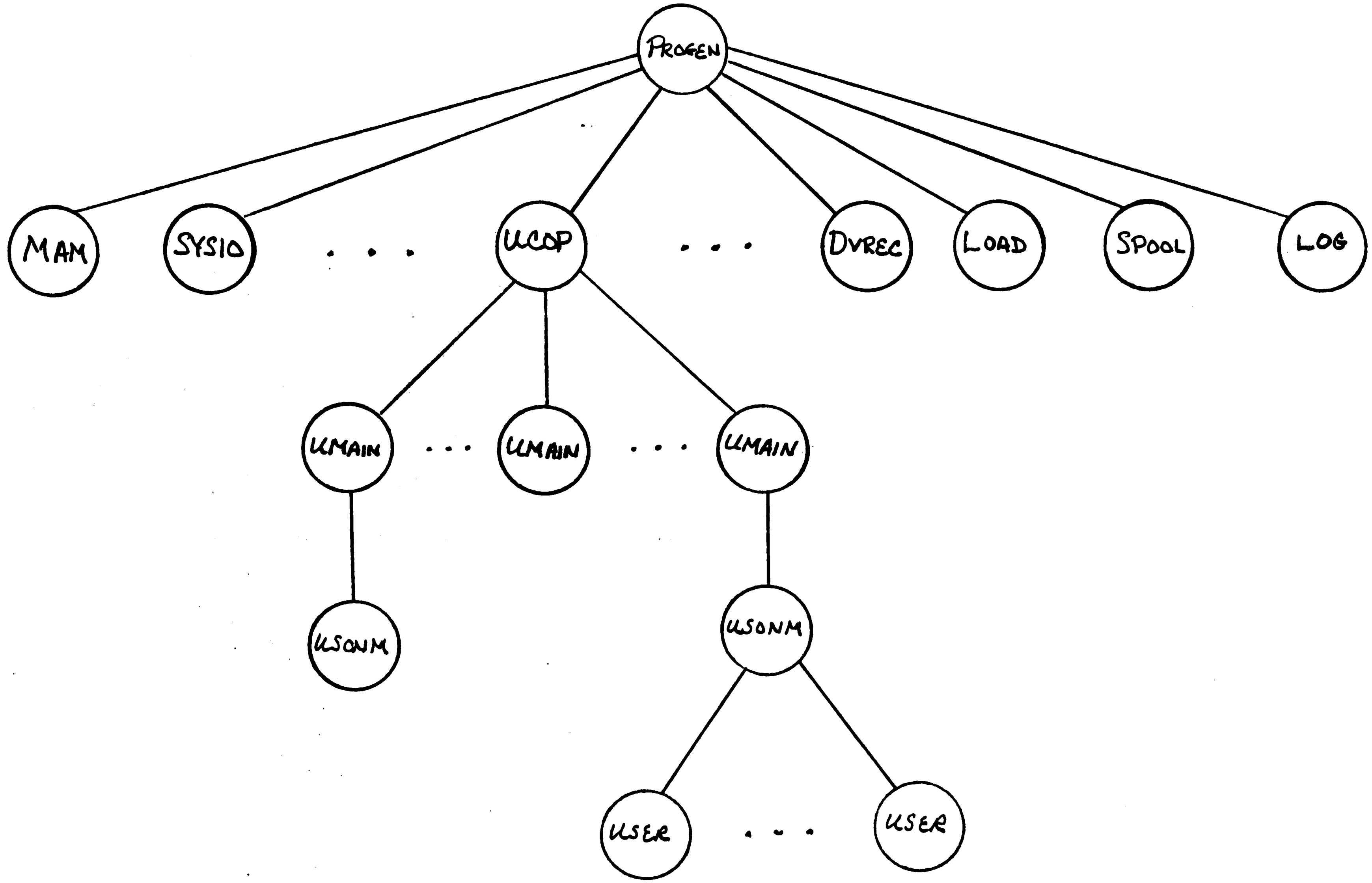
- A CODE SPACE
- A DATA SPACE
  - o PRIVATE, UNSHAREABLE (STACK)
  - o POSSIBLY SHAREABLE (EXTRA DATA SEGS)
- CONTROL STRUCTURES
  - o PCB (CORE RESIDENT)
  - o PCBX (NON-RESIDENT)

A PROCESS MAY HAVE OTHER RESOURCES AS WELL (RINS, FILE/DEVICES, OTHER PROCESSES)

STRUCTURE OF A PROCESS



PROCESS HIERARCHY IN MPE-III



PROCESS STATE

- MAJOR STATE

o ACTIVE	PROCESS IS RUNNING OR READY TO RUN
o WAIT	SOME EVENT MUST OCCUR BEFORE PROCESS CAN RUN

- ADDITIONAL MODES

o "SHORT" WAIT	DISC TRANSFER
o "LONG" WAIT	I/O, FATHER, SON, RIN, MAIL, MOURNING, JUNK
o IMPEDED	REALLY JUST A WAIT, BUT HOPEFULLY SHORTER
o CRITICAL	PROCESS MUST NOT BE ABORTED
o SIR	PROCESS HAS A "SYSTEM INTERNAL RESOURCE"

PROCESS ACTIVITIES

IN THE COURSE OF ITS EXISTENCE, A PROCESS UNDERGOES SEVERAL ACTIVITIES

- o CREATION
- o TERMINATION
- o COMMUNICATION
- o SYNCHRONIZATION

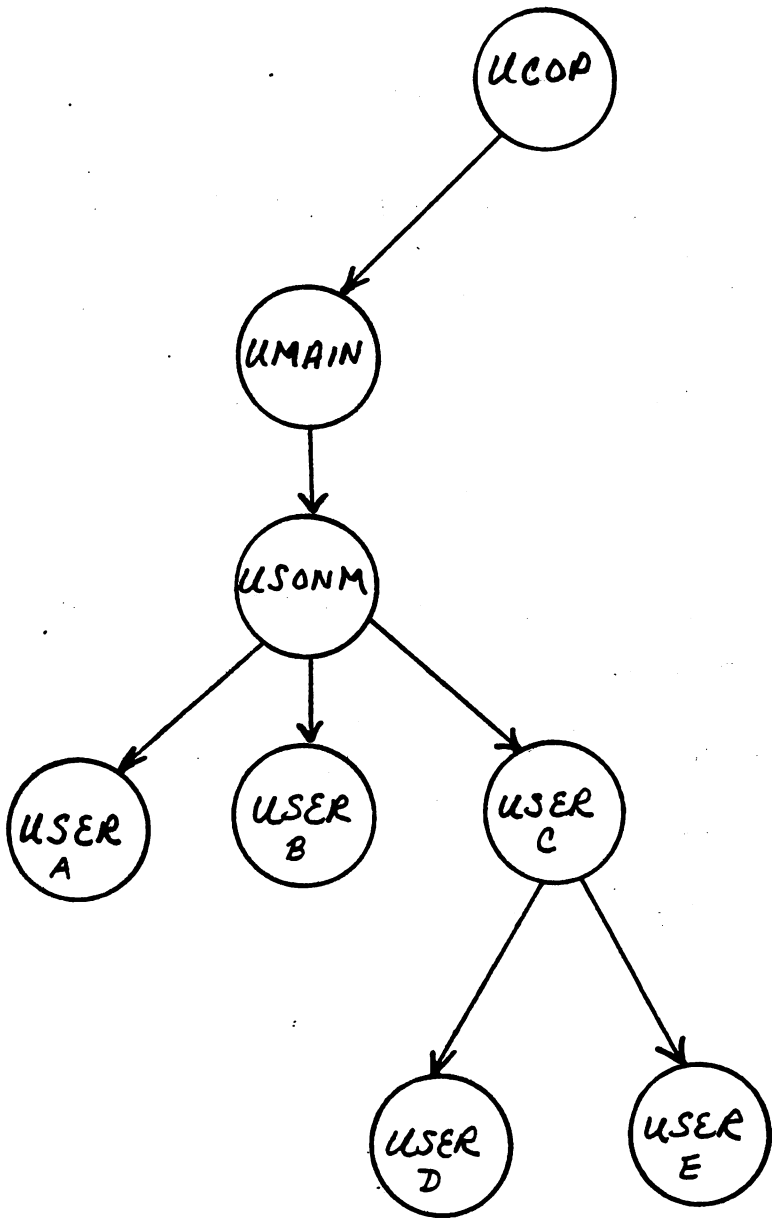
PROCESS CREATION

- GET A PCB FOR THE NEW PROCESS
- LINK PROGRAM FILE IN VIRTUAL MEMORY
- GET A DAT SEGMENT FOR THE NEW STACK
- FORMAT THE PCBX
- FORMAT PCB AND LINK INTO STRUCTURE
- MARK PCB AS BEING IN FATHER WAIT

PROCESS TERMINATION

- GENERAL PHILOSOPHY: LET A PROCESS GIVE UP AS MANY OF ITS RESOURCES AS POSSIBLE.
- HAVE SOME OTHER PROCESS REMOVE STACK AND PCB OF TERMINATING PROCESS.
- ALGORITHM:
  - o LET PROCESS RETURN "NON-ESSENTIAL" RESOURCES TO SYSTEM (RINS, FILES, ETC.)
  - o "STOP" SONS FROM RUNNING AND FORCE THEM TO START TERMINATE SEQUENCE.
  - o WAIT FOR EACH SON TO "DIE" AND THEN RETURN ITS STACK AND PCB.
  - o ASK FATHER OR UCOP TO TAKE AWAY THIS STACK AND PCB.

PROCESS DELETION EXAMPLE



PROCESS COMMUNICATION

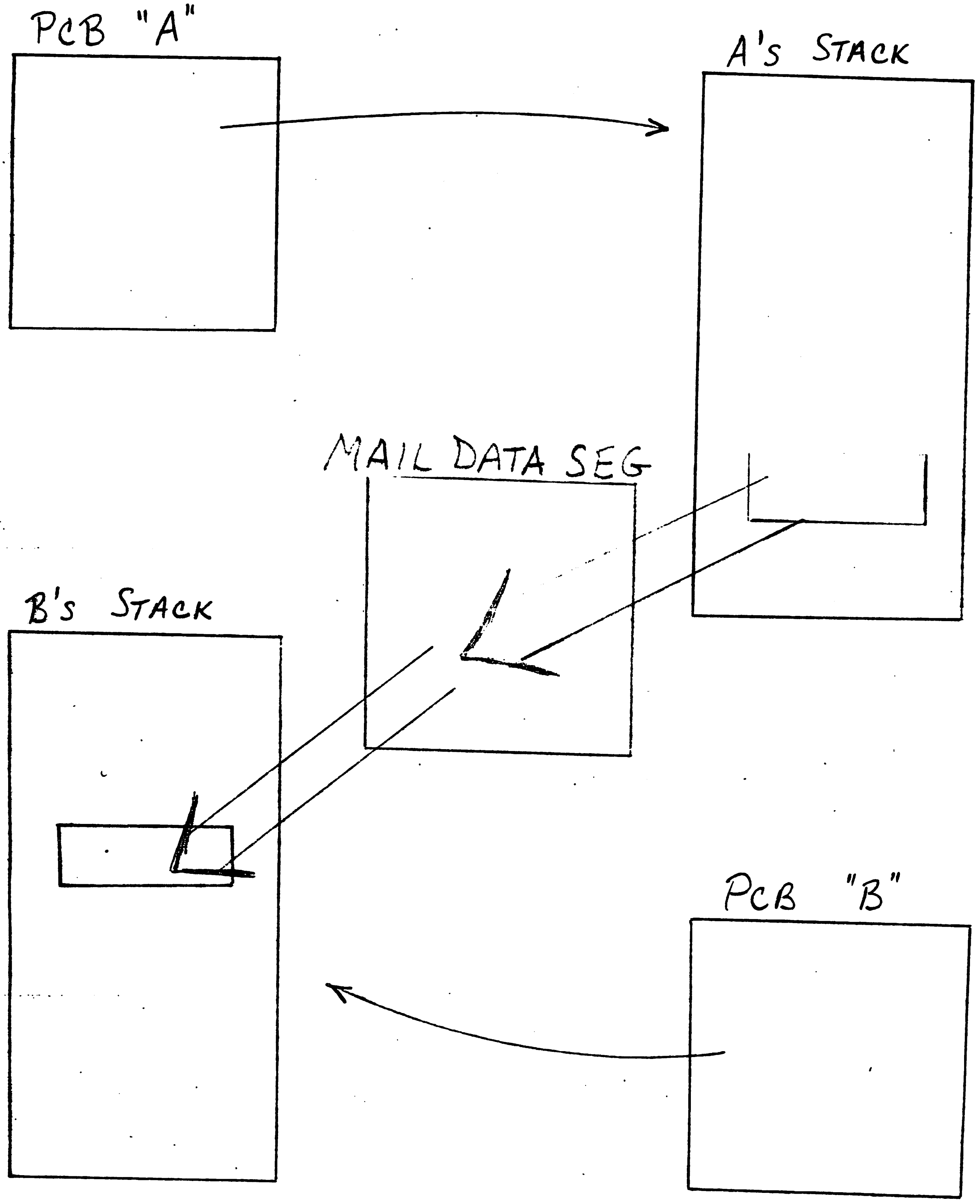
- A PROCESS MAY COMMUNICATE WITH OTHER PROCESSES THROUGH A VARIETY OF MEANS.
  - o SHARED FILE
  - o EXTRA DATA SEGMENT
  - o MAIL FACILITY

- A PROCESS MAY COMMUNICATE WITH ITS FATHER OR 1 OF ITS SONS THROUGH A "MAILBOX".

- BETWEEN ANY TWO PROCESSES, THERE CAN BE ONLY 1 OUTSTANDING MESSAGE AT A TIME.

- MAIL MUST BE "SENT" AND "RECEIVED".

MAIL MECHANISM

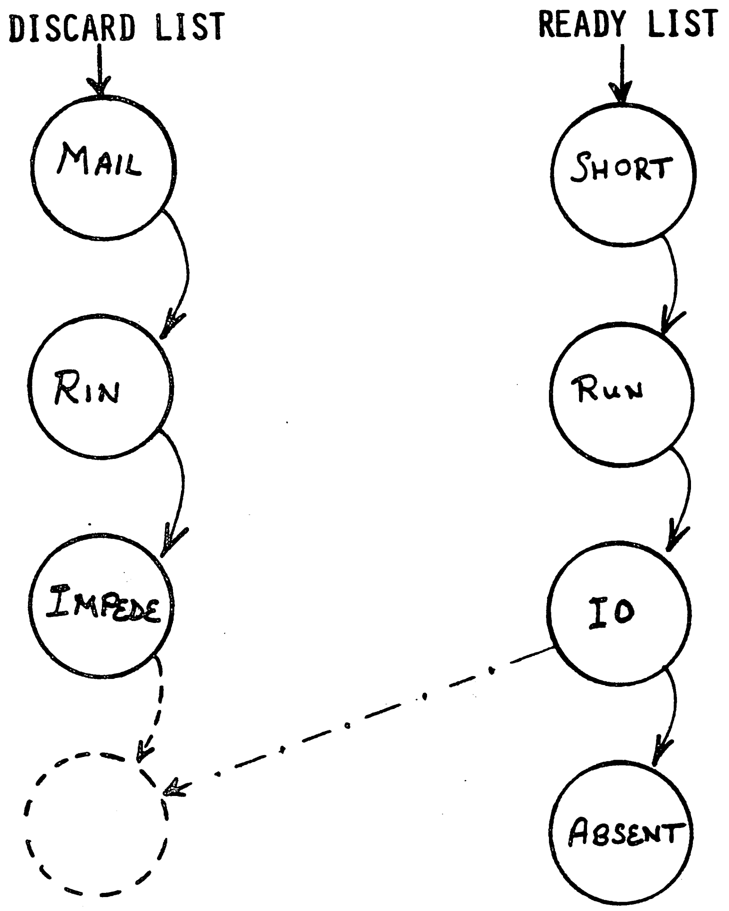


PROCESS SYNCHRONIZATION

A PROCESS MAY CAUSE ANOTHER PROCESS TO BECOME ACTIVE OR IT MAY CAUSE ITSELF TO BECOME SUSPENDED.

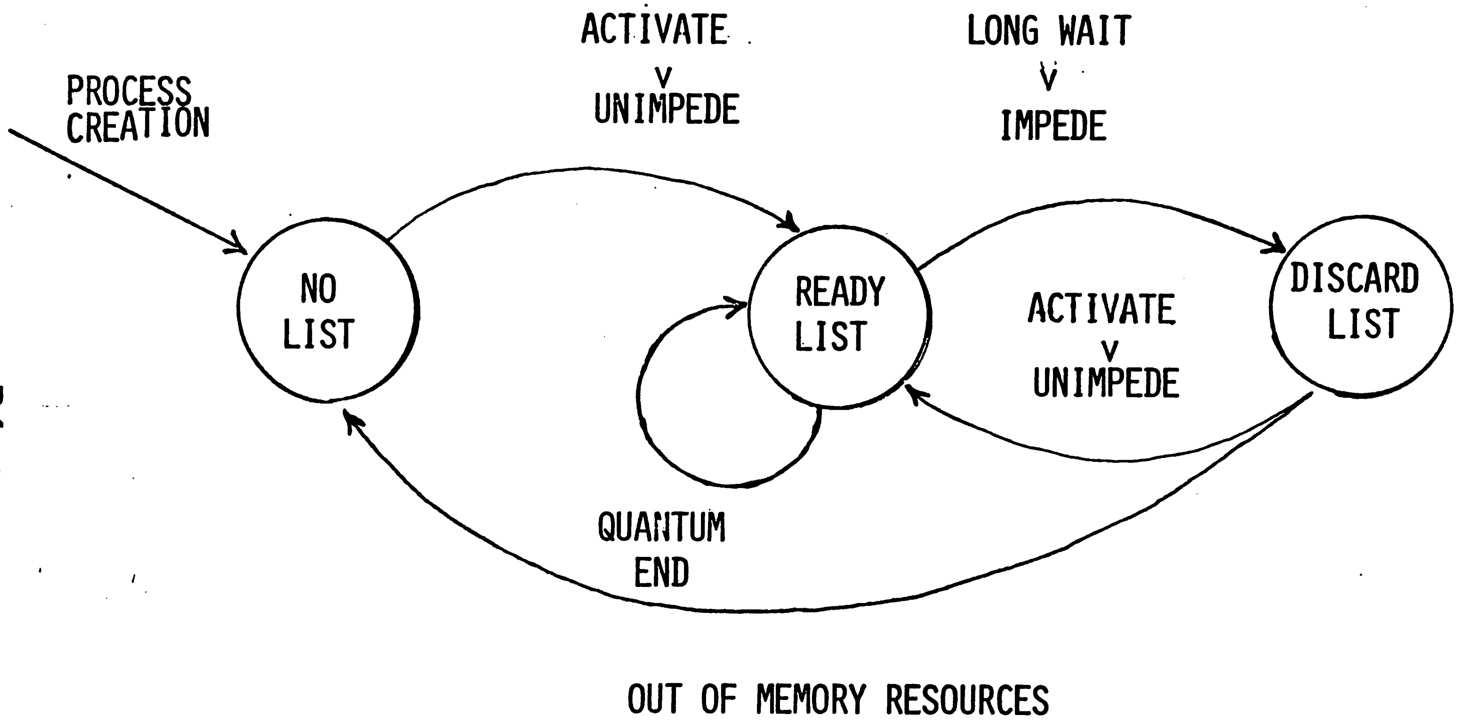
```
PROCEDURE ACTIVATE (P);
   BEGIN
     MARK P'S PCB AS "ACTIVE";
     LINE PCB P INTO READY LIST;
   END;


PROCEDURE SUSPEND;
   BEGIN
     MARK MY PCB AS "WAITING FOR EVENT";
     EXECUTE DISP INSTRUCTION;
   END;
```

DISPATCHER'S SCHEDULING QUEUES



- READY LIST	ALL PROCESSES WHICH WANT TO USE THE CPU
- DISCARD LIST	PROCESSES WHICH CAN'T USE CPU NOW AND HAVE MEMORY.

HOW PROCESSES MOVE BETWEEN LISTS



DYNAMIC PRIORITIES AND READY LIST

- PRIORITIES ARE INTEGERS BETWEEN 1 AND 255
  (1 IS HIGHEST; 255 IS LOWEST)
- READY LIST IS ORDERED BY PRIORITY AND MANAGED AS A SET OF SUBQUEUES.
- A SUBQUEUE IS A RANGE OF PRIORITIES AND MANAGED LINEARLY OR CIRCULARLY.
- PROCESSES IN A AND B QUEUES HAVE FIXED PRIORITIES.
- PROCESSES IN C, D, AND E QUEUES HAVE DYNAMIC PRIORTIES.

MOVEMENT RULES FOR C, D, AND E

- INCREASE RULE: IF PROCESS X CUTS IN FRONT OF PROCESS Y THEN INCREASE Y'S PRORITY BY 1 (I.E. $PRI_Y := PRI_Y - 1$)

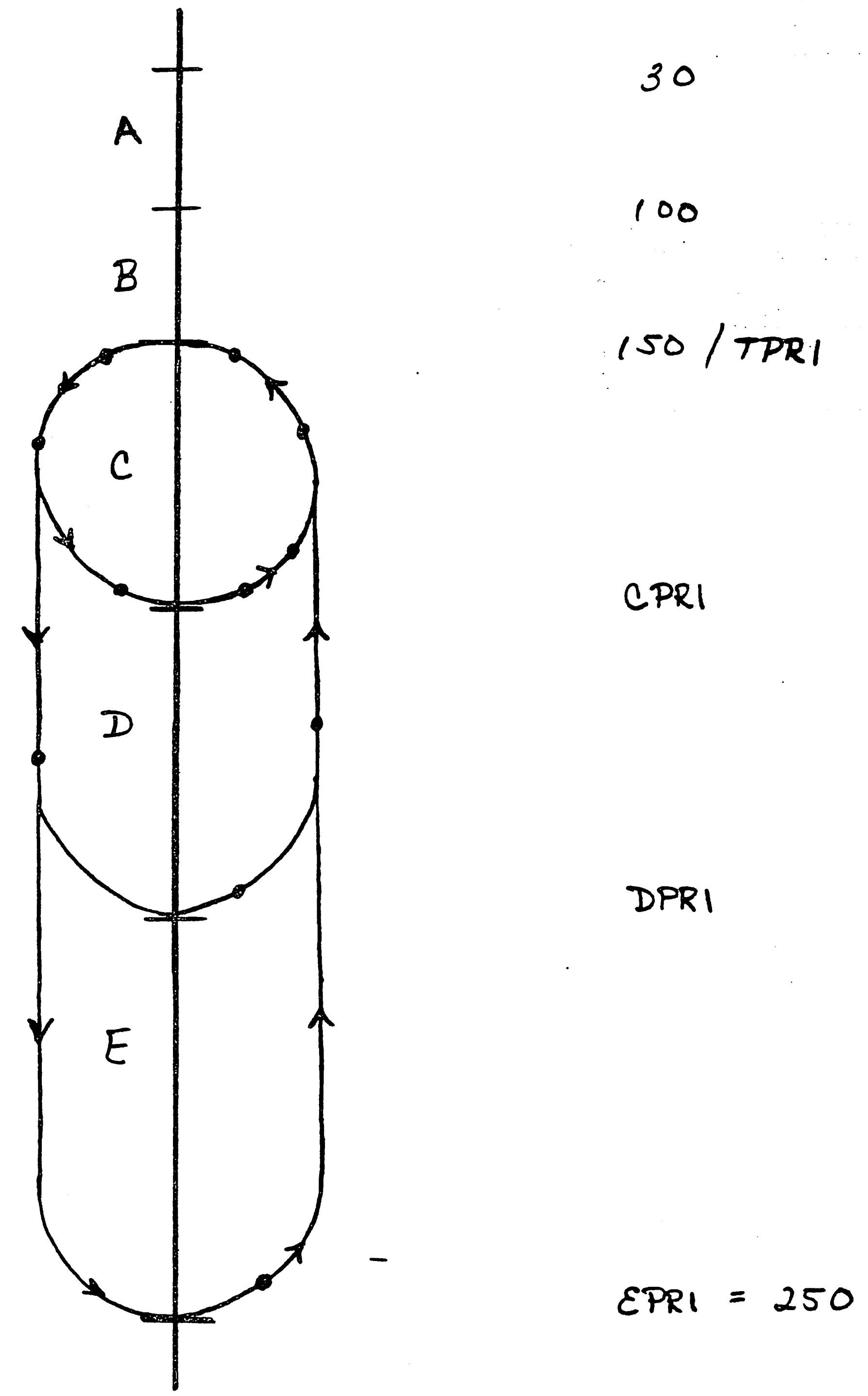
- DECREASE RULE: AFTER COMPLETION OF AN ENTIRE QUANTUM

FOR CS ----> NEWPRI := OLDPRI + 1
(BUT NO LOWER THAN CPR1)

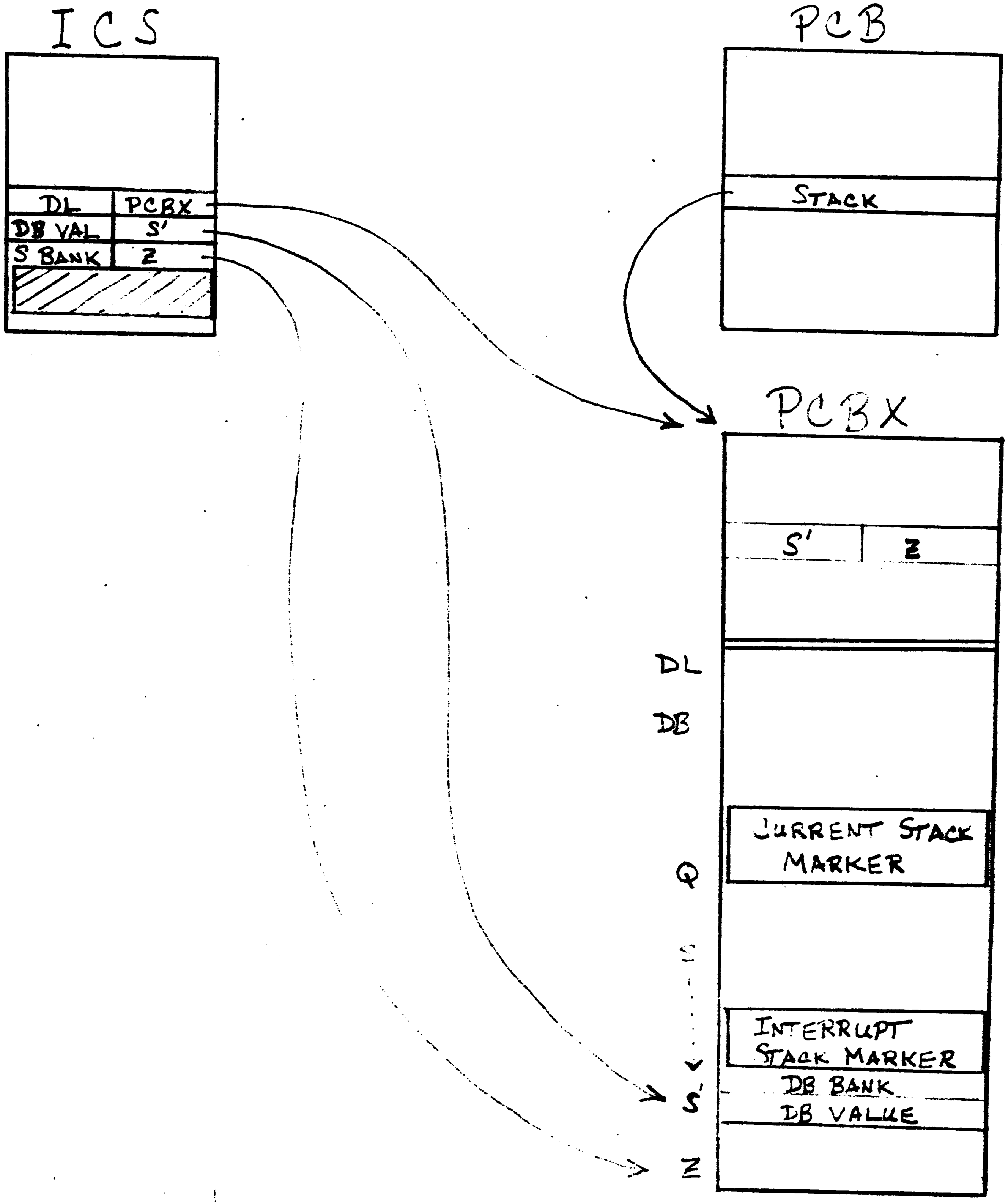
FOR DS ----> NEWPRI := (((OLD PRI - 150) +1) * 3) + 150
(BUT NO LOWER THAN DPRI)

FOR ES ----> NEWPRI := 250 <u>ALWAYS</u>

READY LIST AS SUBQUEUES



PROCESS QUIESCE



SIMPLE DISPATCHING

DISPATCHER IS THE SYSTEM ROUTINE WHICH DETERMINES THE FAIR ALLOCATION OF THE CPU AMONG THE CURRENTLY ACTIVE PROCESSES.

```
PROCEDURE DISPATCHER;
  BEGIN
    << QUIESCE THE RUNNING PROCESS >>
    UPDATE S-REGISTER SETTING IN PCBX;
    IF USER-LEVEL PROCESS THEN UPDATE CPU USE;
    IF CIRCULARLY-SCHEDULED AND QUANTUM OVER THEN
       COMPUTE NEW PRIORITY;

    << LAUNCH THE NEXT READY PROCESS >>
    SCAN READY LIST FOR HIGHEST PRIORITY PROCESS;
    READ REGISTER SETTINGS PCBX ----> ICS;
    SET TIMER WITH PROCESS'S QUANTUM;
    IXIT TO START PROCESS EXECUTION;
  END;
```

COMPLICATED DISPATCHING

- HIGHEST PRIORITY, READY PROCESS MAY NOT HAVE ALL REQUIRED SEGMENTS PRESENT IN MAIN MEMORY.

- DISPATCHER DETECTS THE PROCESS IS OUT OR ABSENT AND SCHEDULES MEMORY MANAGER FOR EXECUTION.

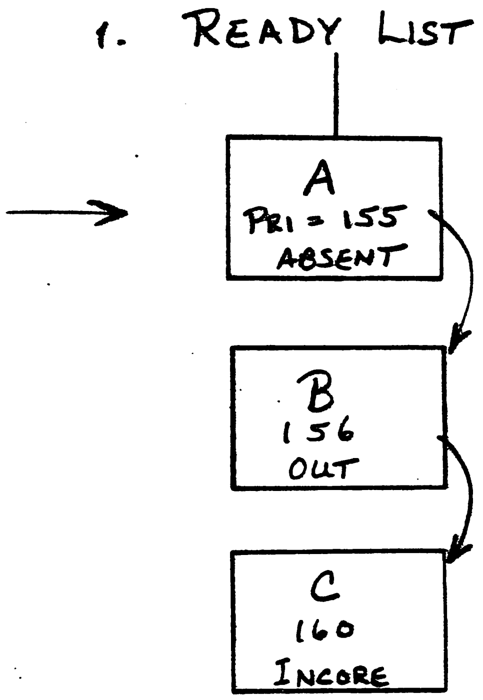
- MAM EXECUTES, SATISFYING THE REQUEST
  - o FIND REAL MEMORY SPACE
  - o INITIATE READ AND SHORT WAIT
  - o COMPLETE REQUEST; MARK PROCESS INCORE

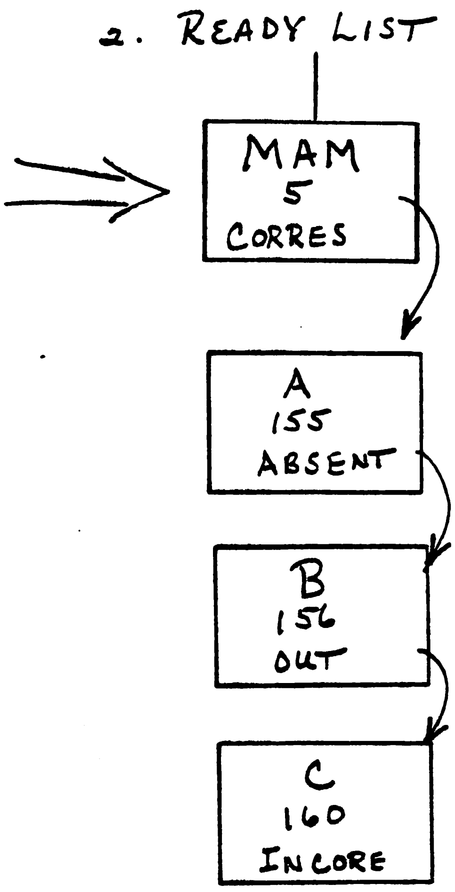
- DISPATCHER NOW FINDS HIGH PRIORITY PROCESS INCORE AND LETS IT RUN.

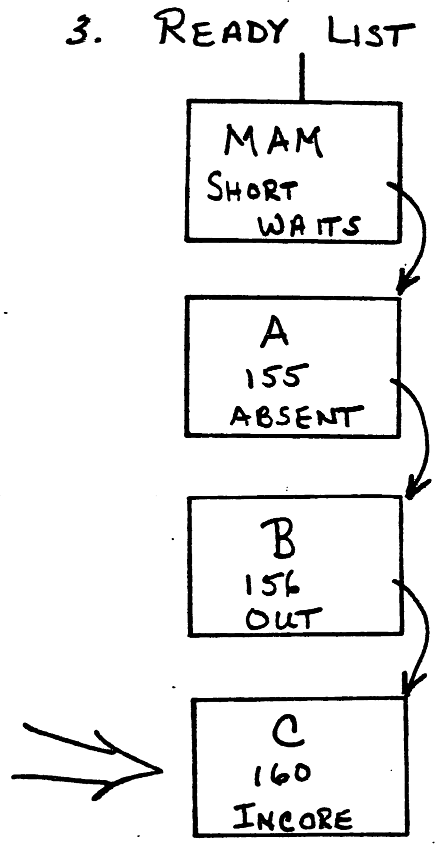
- WHILE MAM IS WAITING/WORKING, THE NEXT HIGHEST PRIORITY, <u>INCORE</u> PROCESS WILL BE RUN.

- SIDE NOTE: IF HIGHEST PRIORITY PROCESS EVER SHORT WAITS, ONLY <u>INCORE</u> LOWER PRIORITY PROCESSES MAY BE RUN.

DISPATCHING EXAMPLE

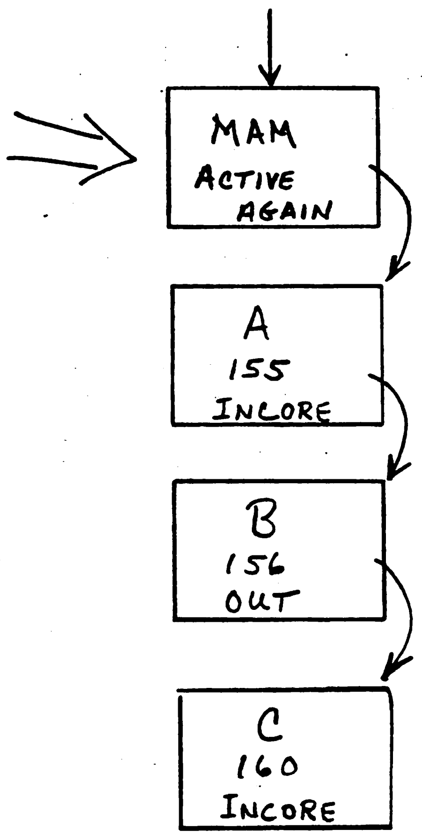




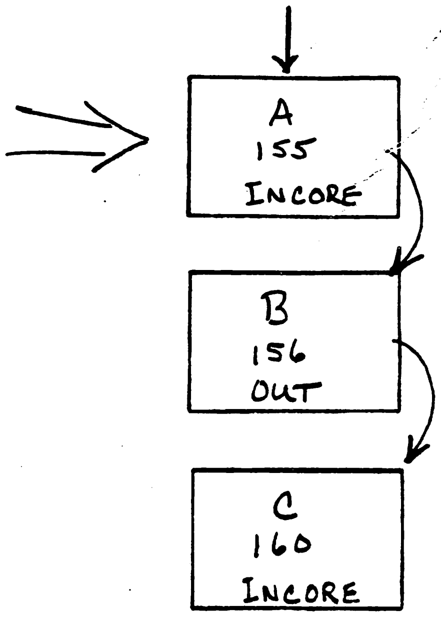


DISPATCHING EXAMPLE

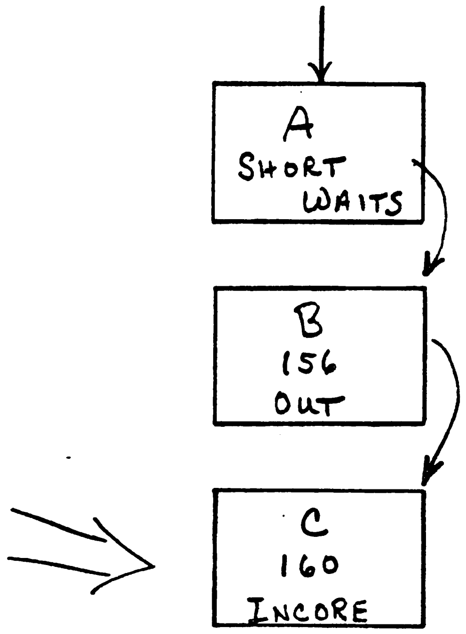
4. READY LIST



5. READY LIST

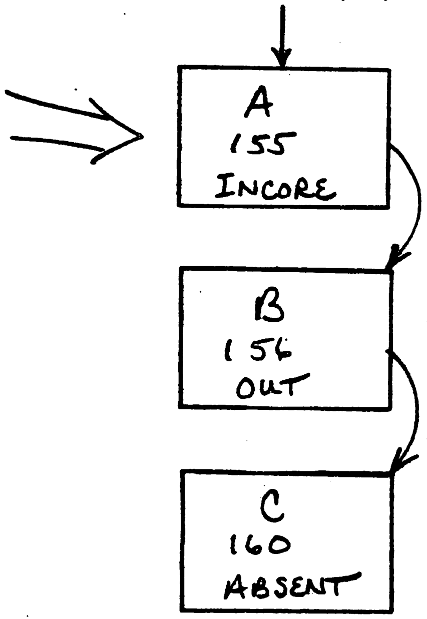


6. READY LIST

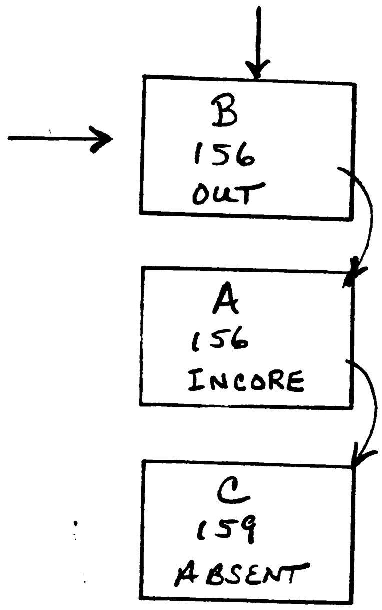


DISPATCHING EXAMPLE

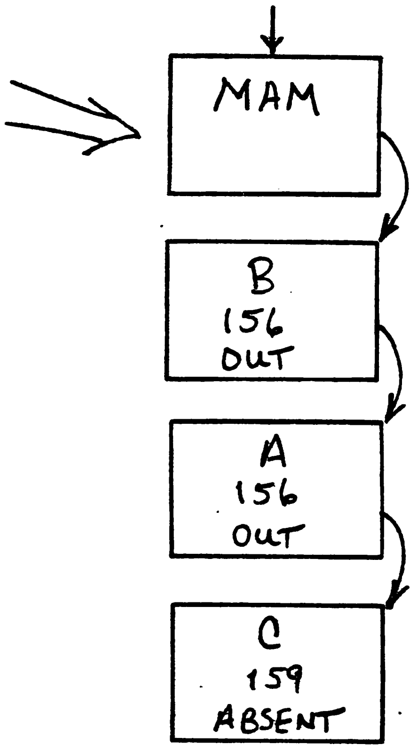
7. READY LIST



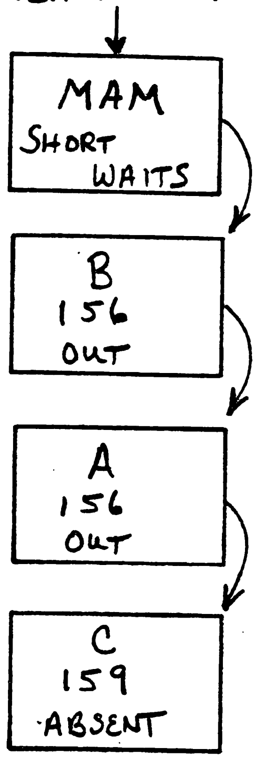
8. READY LIST



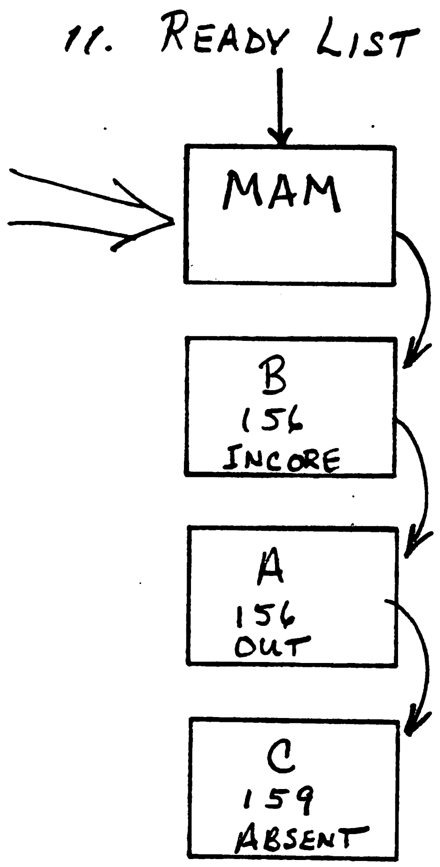
9. READY LIST

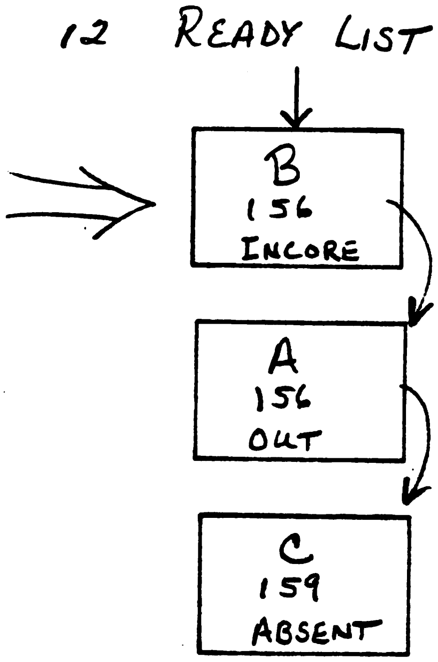


10. READY LIST



DISPATCHING EXAMPLE


11. READY LIST
MAM
B
156
INCORE
A
156
OUT
C
159
ABSENT


12 READY LIST
B
156
INCORE
A
156
OUT
C
159
ABSENT