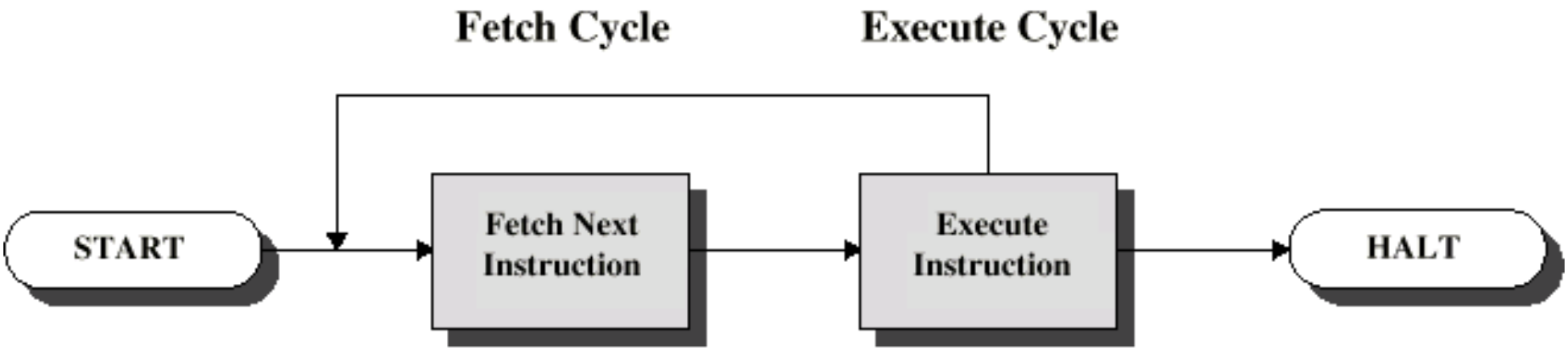
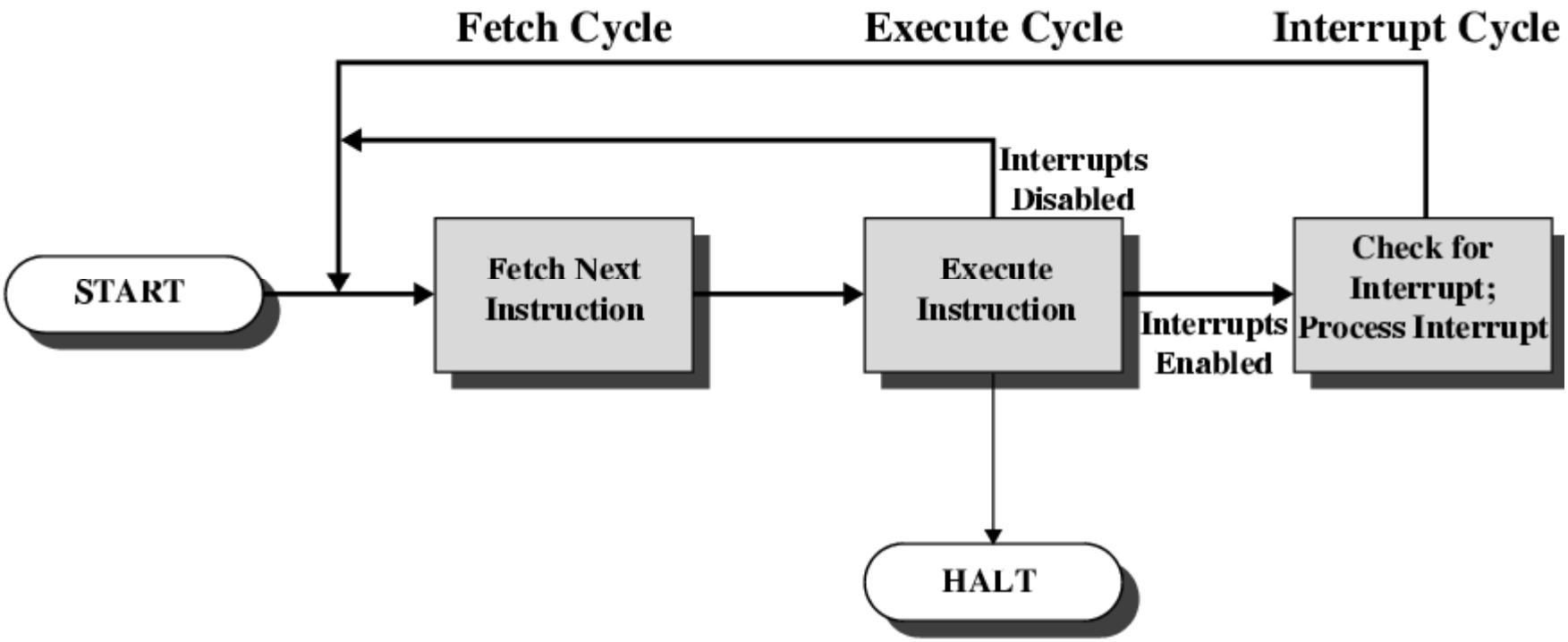
Procedure and Device Service Routine

Instruction Cycle

- Two steps:
  - Fetch
  - Execute



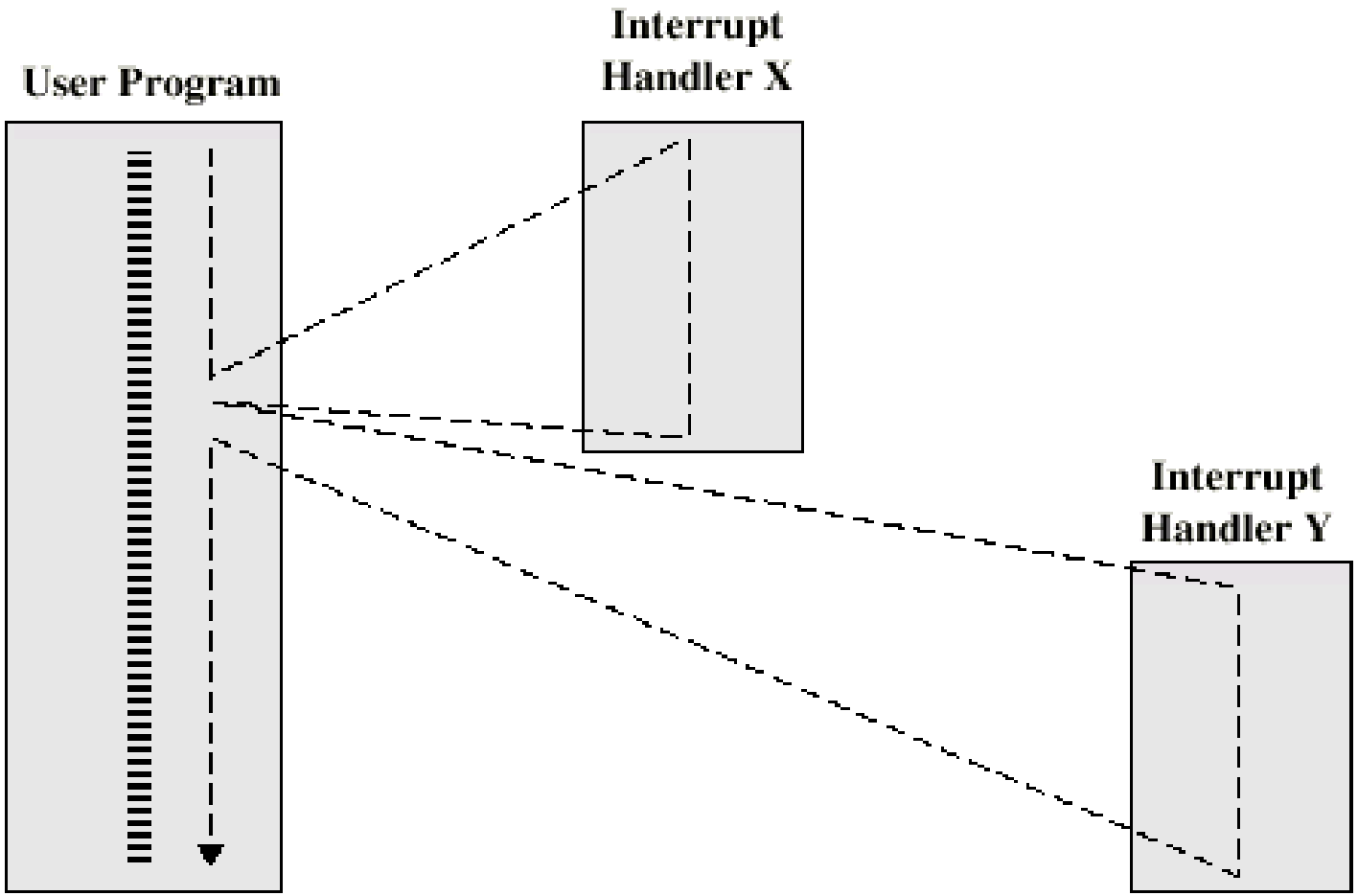
Instruction Cycle with Interrupts



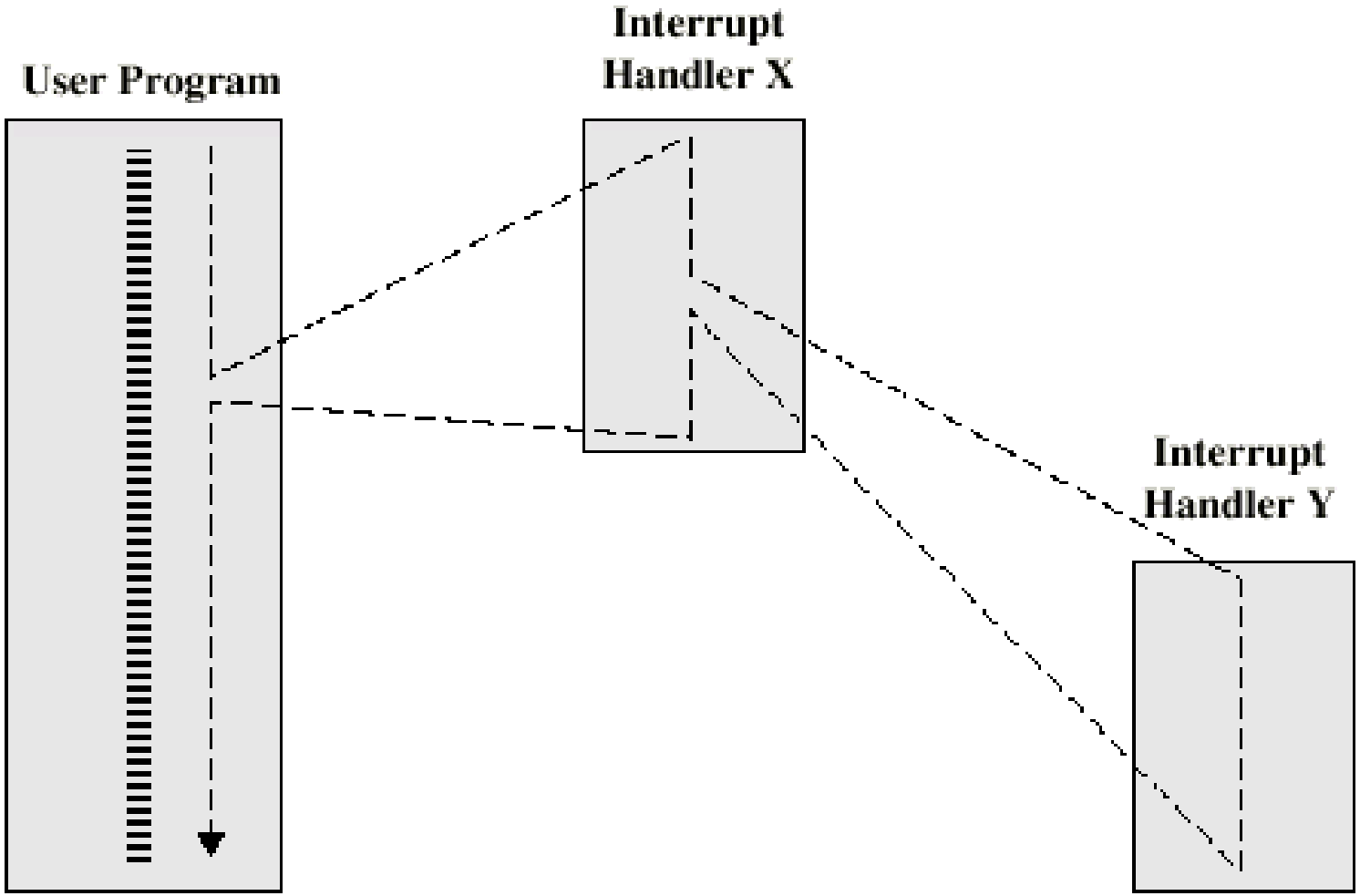
Interrupt Handling

- Interrupt handing routine
  - —A separate program
- Issues related to handle two or more programs at a time

Multiple Interrupts - Sequential



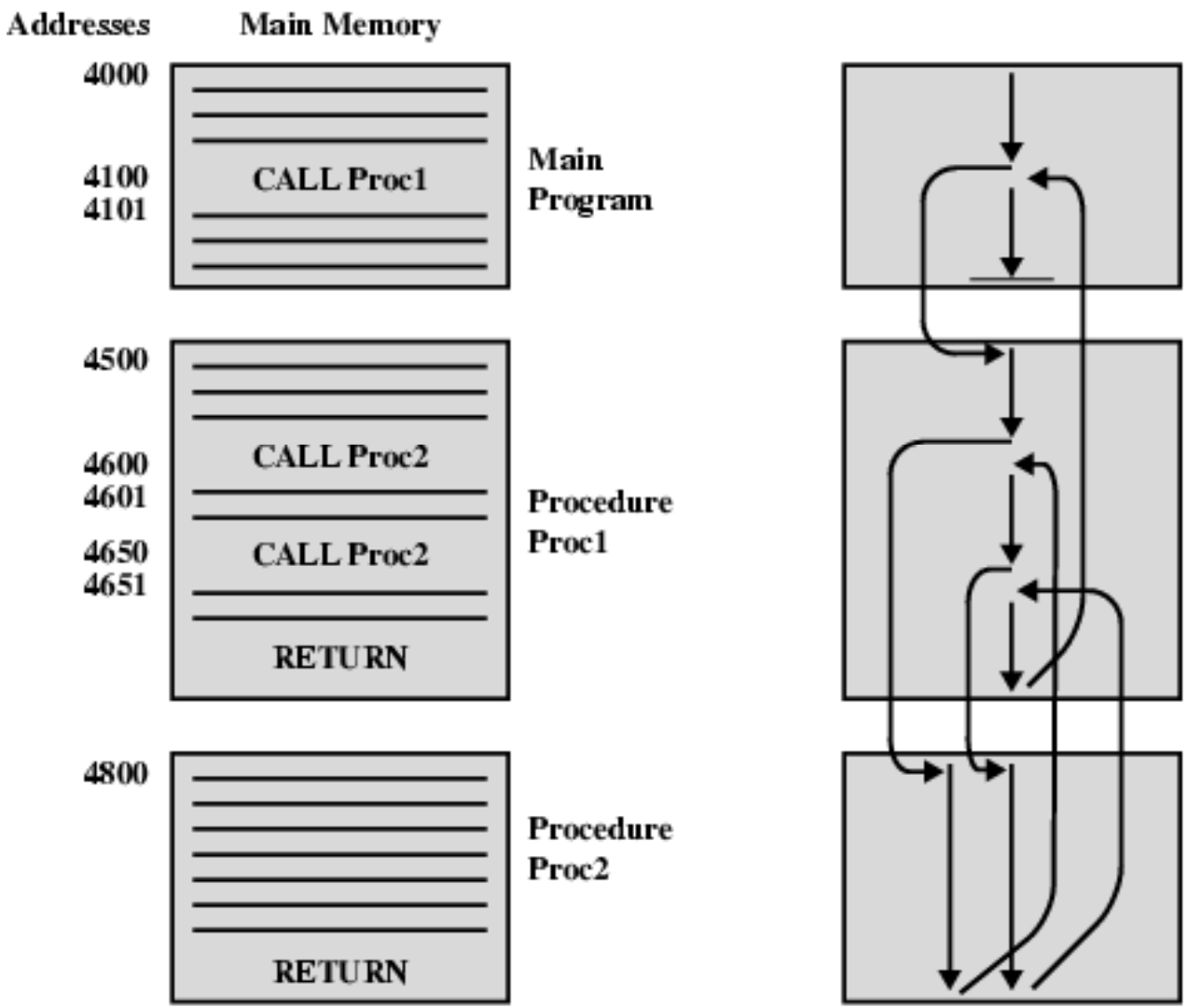
Multiple Interrupts – Nested



Subroutine/Procedure/Function

- Independent unit of code to perform a subtask of the main task.
- Used in modular programming
- How to provide facility for procedure call
- Macros in Programming languages like C

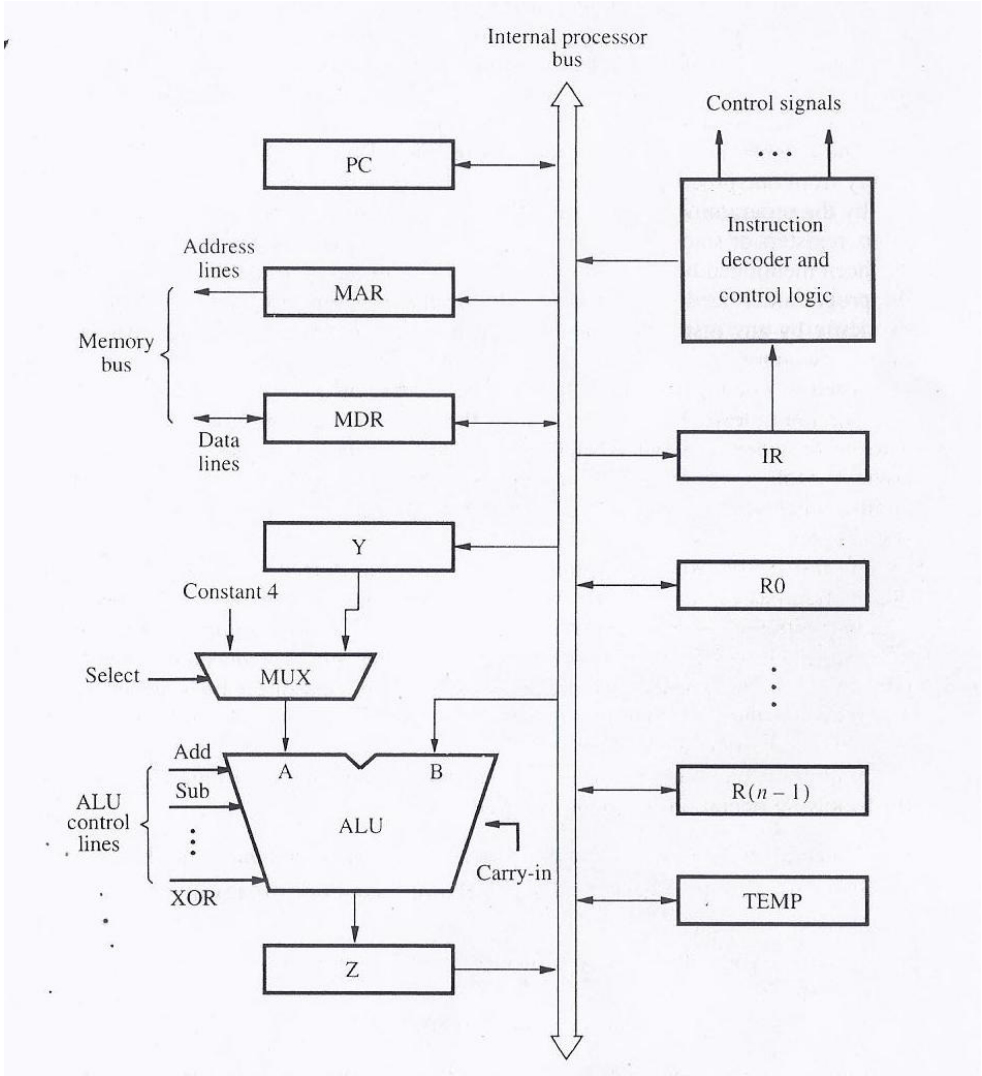
Nested Procedure Calls



Procedure Call

- Tasks to be performed before procedure CALL
  - Retain the current status of the processor
  - After returning from procedure/interrupt routine, we must restart the execution from the point where we have stopped.
- Current status of the processor
  - Program Counter
  - Program Status Word (PSW)
- How to Retain these information
- Any other information need to be saved?

Single Bus Organization



Control Steps: Fetch and Execute

ADD (R3), R1: Add the content of register R1 and memory location pointed by R3; and store the result in R1

Step	Action
1	PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	$R3_{out}$, MAR_{in}, Read
5	$R1_{out}$, Y_{in}, WMFC
6	MDR_{out}, SelectY, Add, Z_{in}
7	Z_{out}, $R1_{in}$, End

Modification in Organization

- Store the relevant information in main memory
  - Implement a stack in MM (Control Stack)
- Need to keep the address where to store
  - Use of a register, SP: Stack Pointer
  - To keep the address of the Top of the Stack
- After completion of the procedure, restore the information from stack

Instructions

- PUSH R
  - source is the register R
- POP R
  - destination is the register R
- CALL address
  - starting address of the procedure
- RETURN

PUSH (Execute)

- PUSH Ri
  - MAR <- SP
  - MDR <- Ri
  - Write
  - SP <- SP - 4

POP (Execute)

- POP Ri
  - SP <- SP +4
  - MAR <- SP
  - Read
  - Ri <- MDR

CALL (Execute)

- CALL
  - MAR <- SP
  - MDR <- PC
  - Write
  - SP <- SP - 4
  - MAR <- SP
  - MDR <- PSW
  - Write
  - SP <- SP - 4
  - PC <- $IR_{address}$

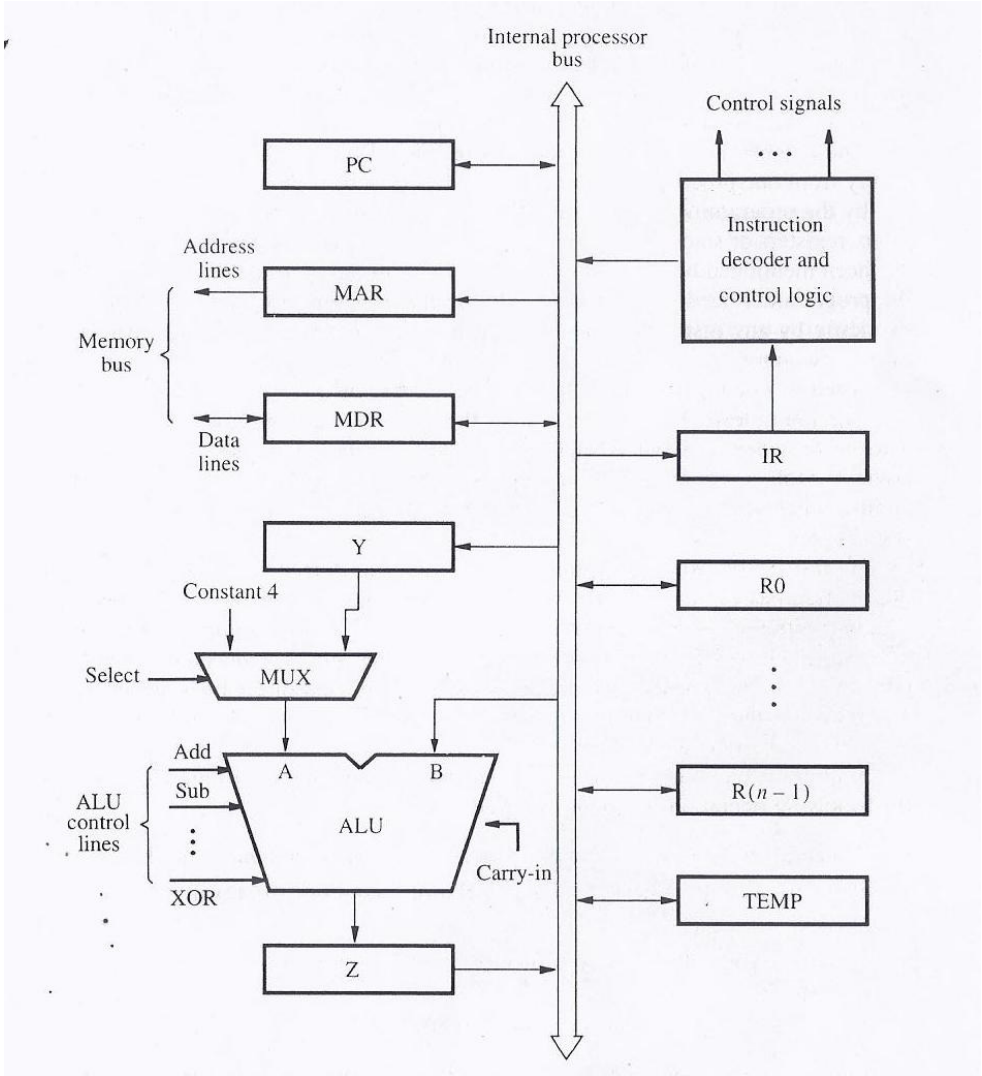
RETURN (Execute)

- RETURN
  - SP <- SP + 4
  - MAR <- SP
  - Read
  - PSW <- MDR
  - SP <- SP + 4
  - MAR <- SP
  - Read
  - PC <- MDR

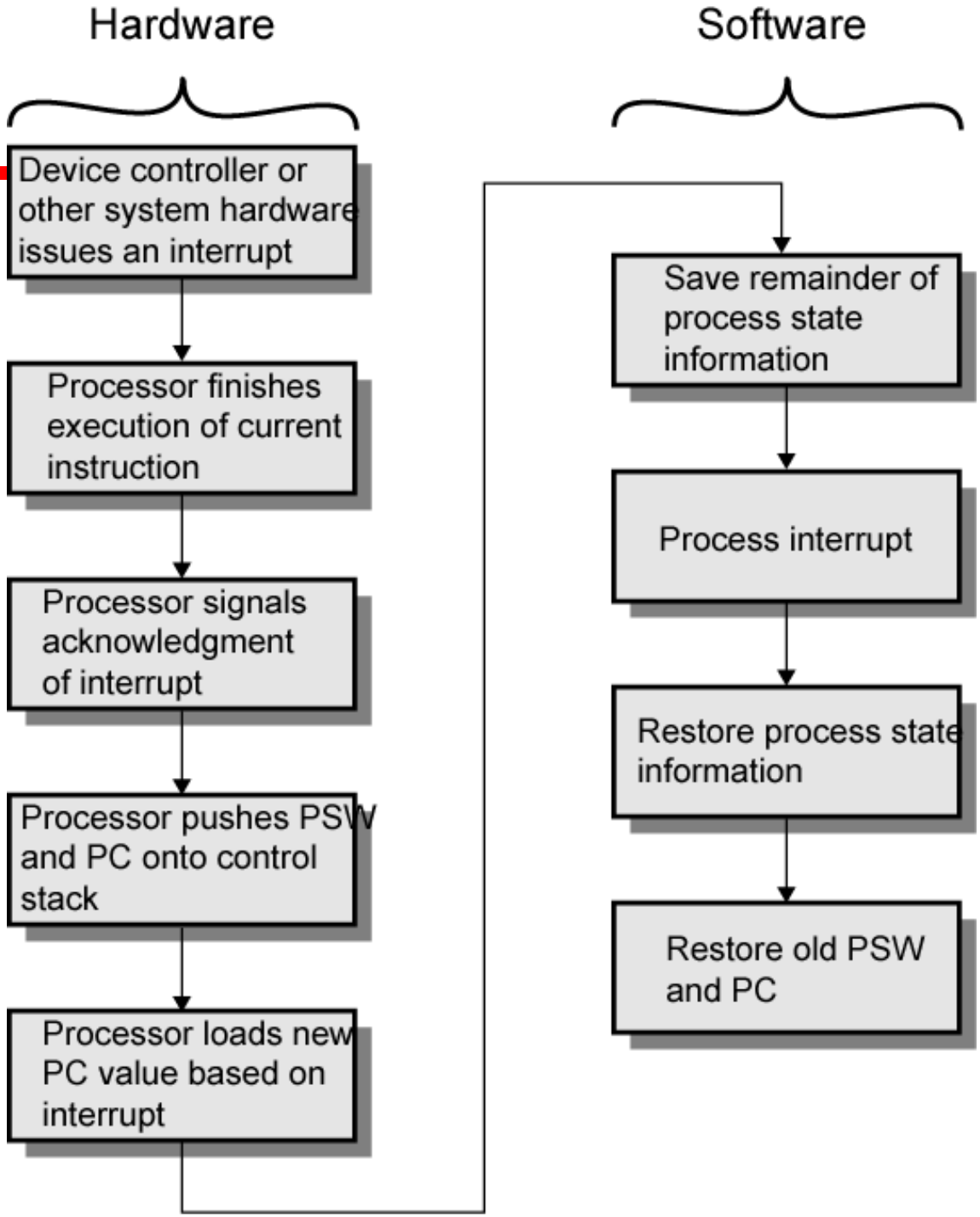
Procedure Call

- Tasks to be performed before procedure CALL
  - Retain the current status of the processor
  - After returning from procedure/interrupt routine, we must restart the execution from the point where we have stopped.
- Current status of the processor
  - Program Counter
  - Program Status Word (PSW)
- How to Retain these information
- Any other information need to be saved?

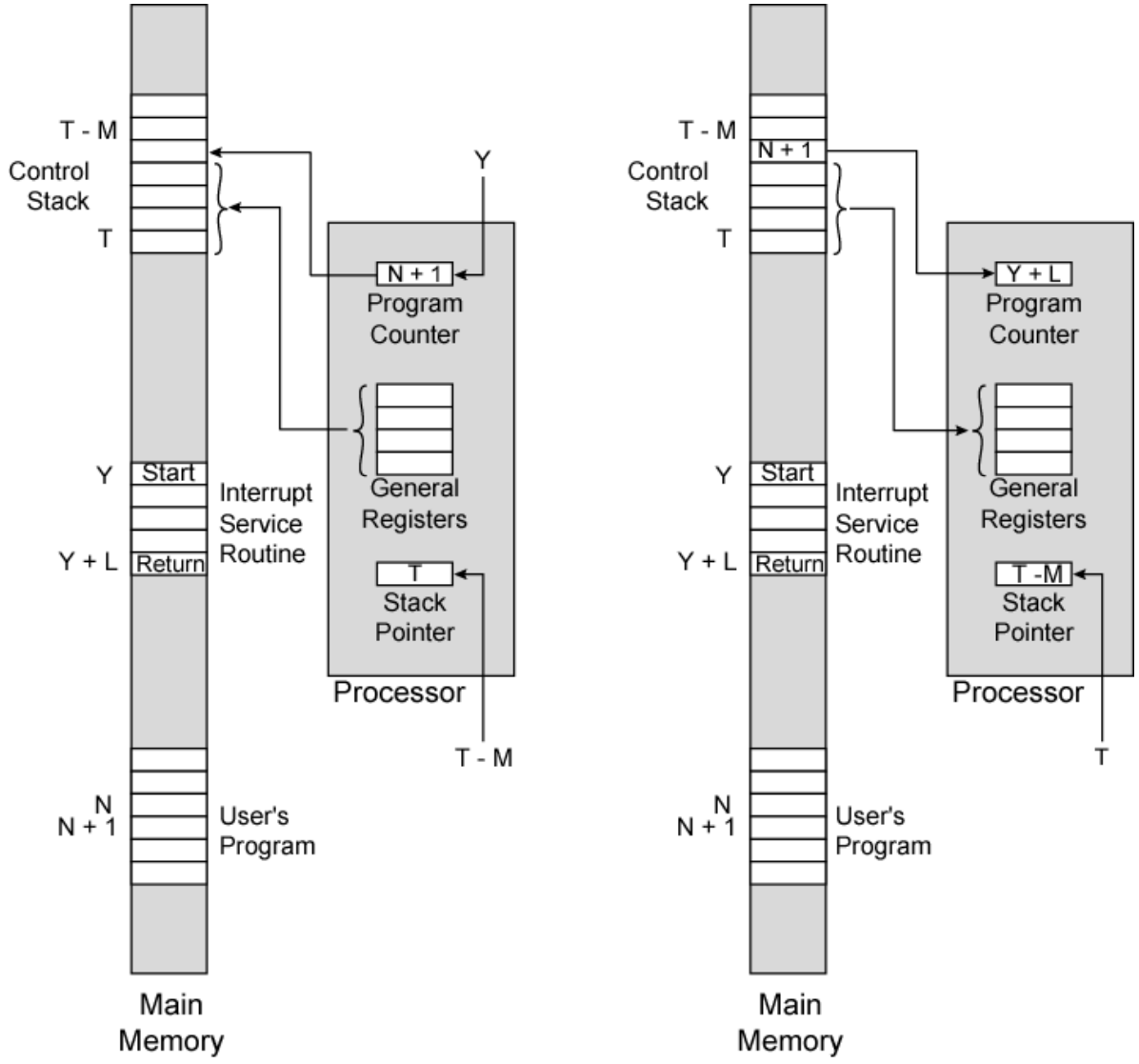
Single Bus Organization



Simple Interrupt Processing



Changes in Memory and Registers for an Interrupt



(a) Interrupt occurs after instruction at location N

(b) Return from interrupt