CS 223 Computer Architecture and Organization

Control Transfer Instructions

J. K. Deka

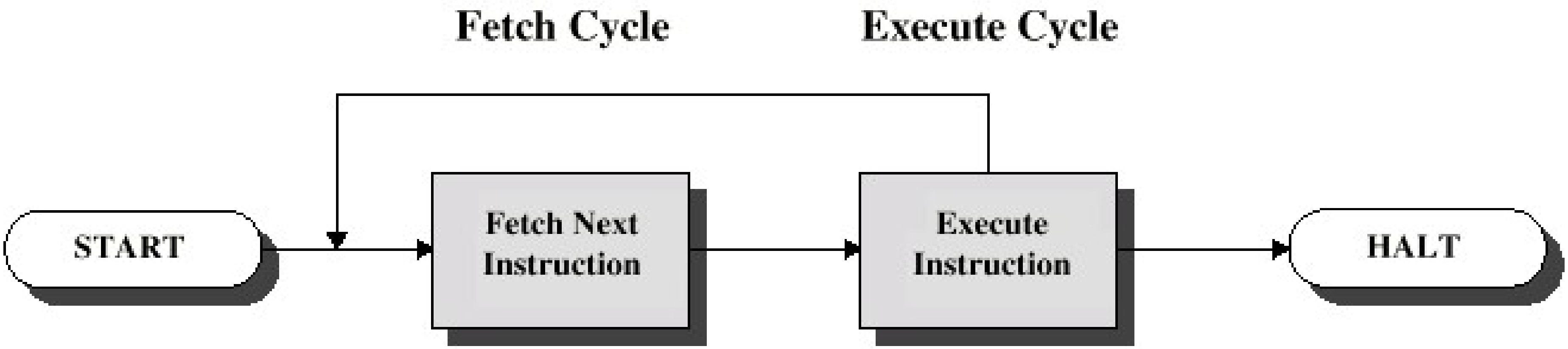
Professor

Department of Computer Science & Engineering

Indian Institute of Technology Guwahati, Assam.

Instruction Cycle

- Two steps:
  - Fetch
  - Execute



Conditional Instructions

- Unconditional Branch (BR)
- Format:

OPCODE	Target Address

Fetch Phase:

t1: MAR <- PC, Read

t2: MBR <- Memory

PC <- PC + 1

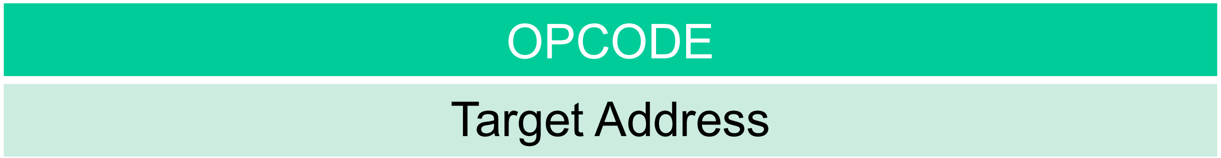
t3: IR <- MBR

Execute Phase:

t4: PC <- $IR_{Address}$

Conditional Instructions

- Unconditional Branch
- Format:



OPCODE

Target Address

Fetch Phase:

t1: MAR <- PC, Read

t2: MBR <- Memory

PC <- PC + 1

t3: IR <- MBR

Execute Phase:

Conditional Instructions

- Conditional Branch (BRZ: Branch on Zero)
- Format:

OPCODE	**Target Address**

Fetch Phase:

t1: MAR <- PC, Read

t2: MBR <- Memory

PC <- PC + 1

t3: IR <- MBR

Execute Phase:

t4: If (Z = 1) PC <- $IR_{Address}$

Conditional Instructions

- Conditional Branch (BRZ: Branch on Zero)
- Format:

OPCODE
Target Address

Fetch Phase:

t1: MAR <- PC, Read

t2: MBR <- Memory

PC <- PC + 1

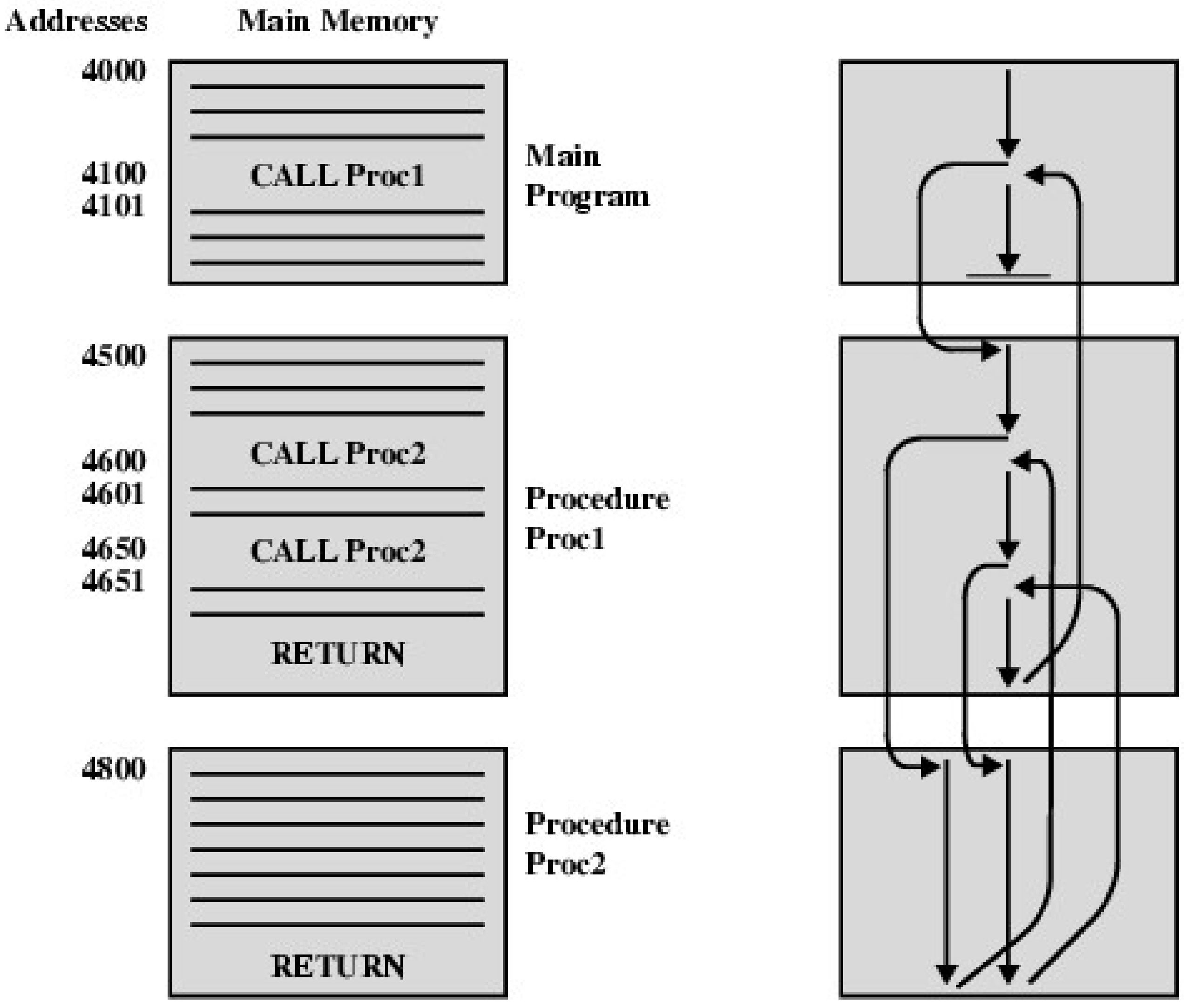
t3: IR <- MBR

Execute Phase:

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
- In case of Interrupt, device service routine is executed

Nested Procedure Calls



Addresses

Main Memory

4000

4100 CALL Proc1

4101

Main Program

4500

4600 CALL Proc2

4601

4650 CALL Proc2

4651

RETURN

Procedure Proc1

4800

RETURN

Procedure Proc2

(a) Calls and returns

(b) Execution sequence

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?

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

- Four different ways for implementation

PUSH (Execute)

- PUSH Ri
  - MAR <- SP
  - MDR <- Ri
  - Write
  - SP <- SP - 1

POP (Execute)

- POP Ri
  - SP <- SP +1
  - MAR <- SP
  - Read
  - Ri <- MDR

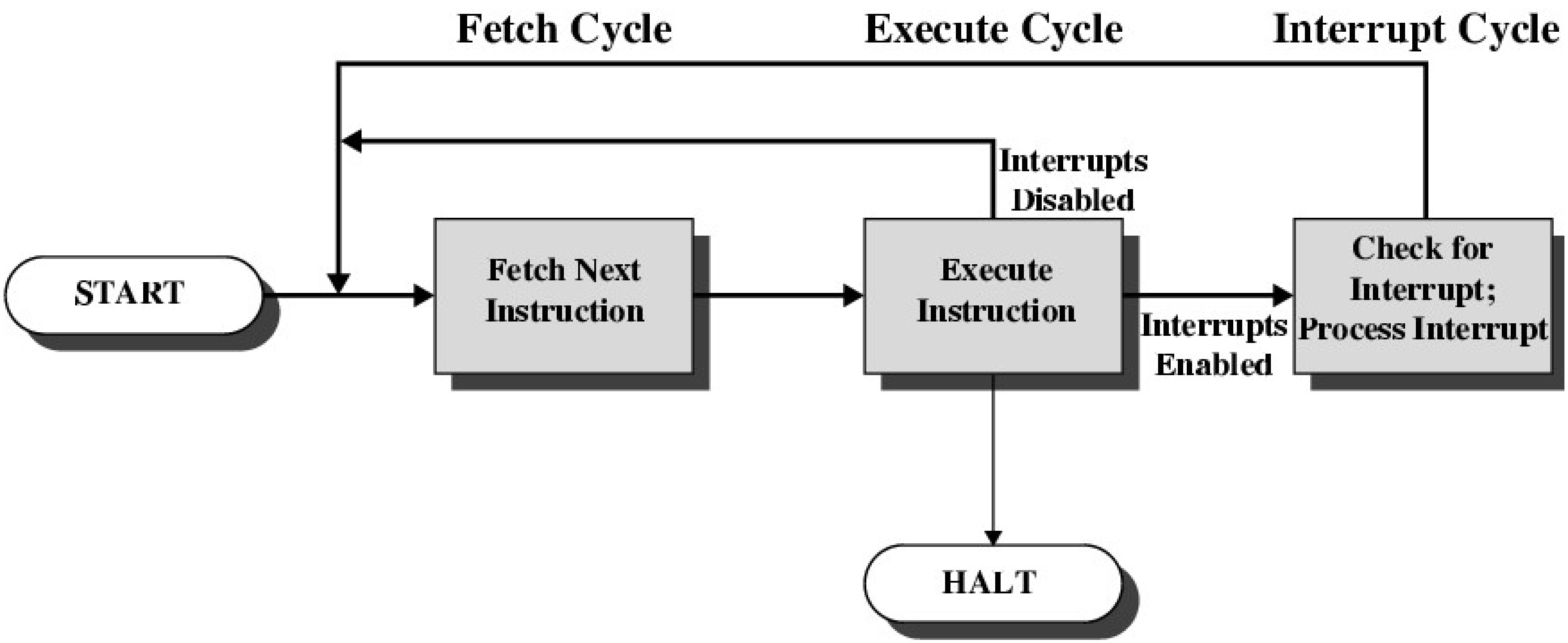
CALL (Execute)

- CALL
  - MAR <- SP
  - MDR <- PC
  - Write
  - SP <- SP -1
  - MAR <- SP
  - MDR <- PSW
  - Write
  - SP <- SP -1
  - PC <- $\mathrm{IR}_{\mathrm{address}}$

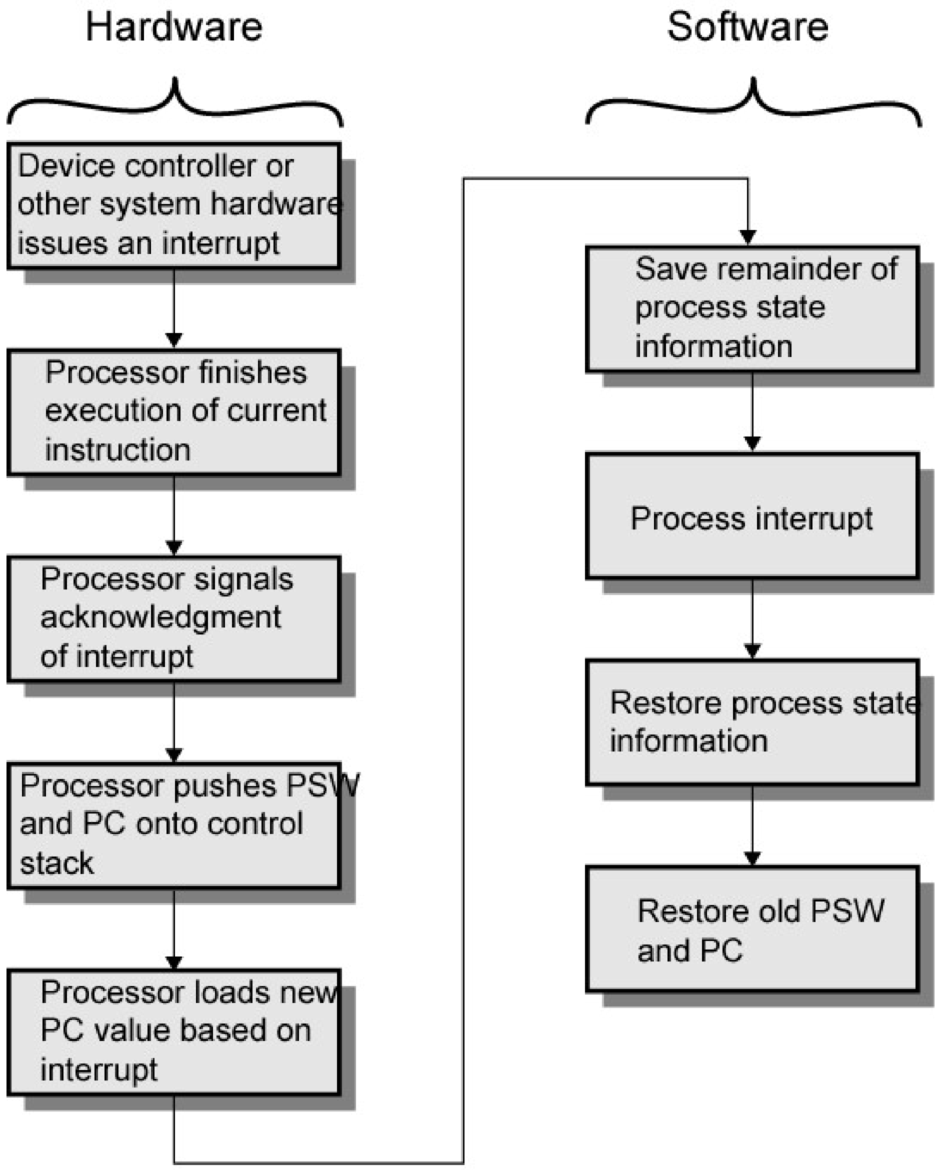
RETURN (Execute)

- RETURN
  - SP <- SP + 1
  - MAR <- SP
  - Read
  - PSW <- MDR
  - SP <- SP + 1
  - MAR <- SP
  - Read
  - PC <- MDR

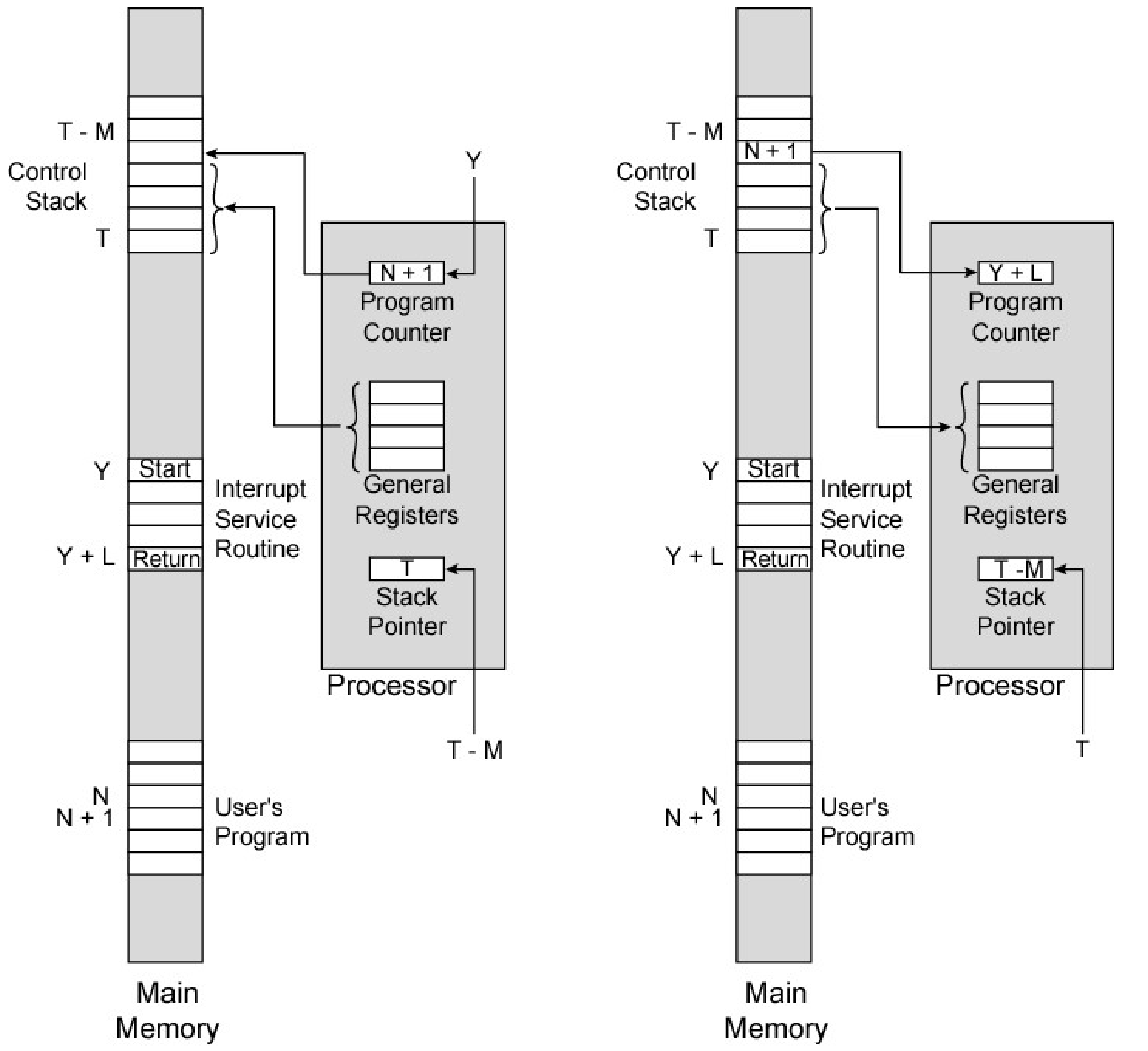
Instruction Cycle with Interrupts



Simple Interrupt Processing


Hardware
Software
Device controller or other system hardware issues an interrupt
Processor finishes execution of current instruction
Processor signals acknowledgment of interrupt
Processor pushes PSW and PC onto control stack
Processor loads new PC value based on interrupt
Save remainder of process state information
Process interrupt
Restore process state information
Restore old PSW and PC

Memory and Registers for an Interrupt



(a) Interrupt occurs after instruction at location N

(b) Return from interrupt