EE321

Computer Architecture

Chap. 03 : Central Processing Unit (CPU)

Dr. Abdelhakim Khouas

Email : akhouas@hotmail.com

ab.khouas@univ-boumerdes.dz

IGEE (ex. INELEC)

University M'hamed Bougara of Boumerdes

Course chapters

1. Review of Digital Design
2. Top level of Computer
3. Central Processing Unit (CPU)
4. Control Unit
5. Memory
6. Instruction Set and Addressing Modes

Lecture Outline

1. CPU Organization
2. Registers Organization
3. Instruction Cycle
4. Interrupts
5. CPU Examples
    1. The X86 Processor Family
    2. The ARM Processor
    3. The Z80 Processor

Readings

Textbook

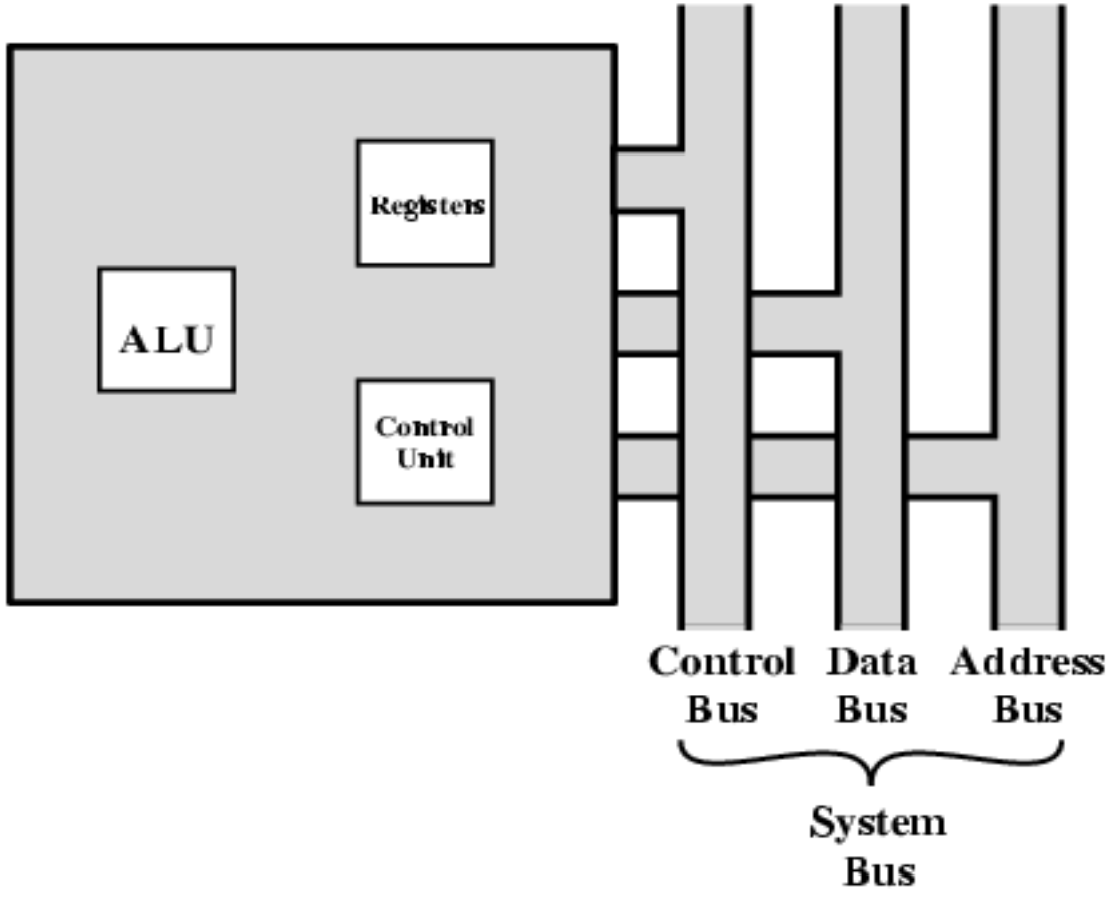
Computer Organization and Structure, Designing for Performance, By William Stallings, 8th edition

Sections

- Chapter 3, sections: 3.1 and 3.2
- Chapter 12, sections: 12.1, 12.2 and 12.3

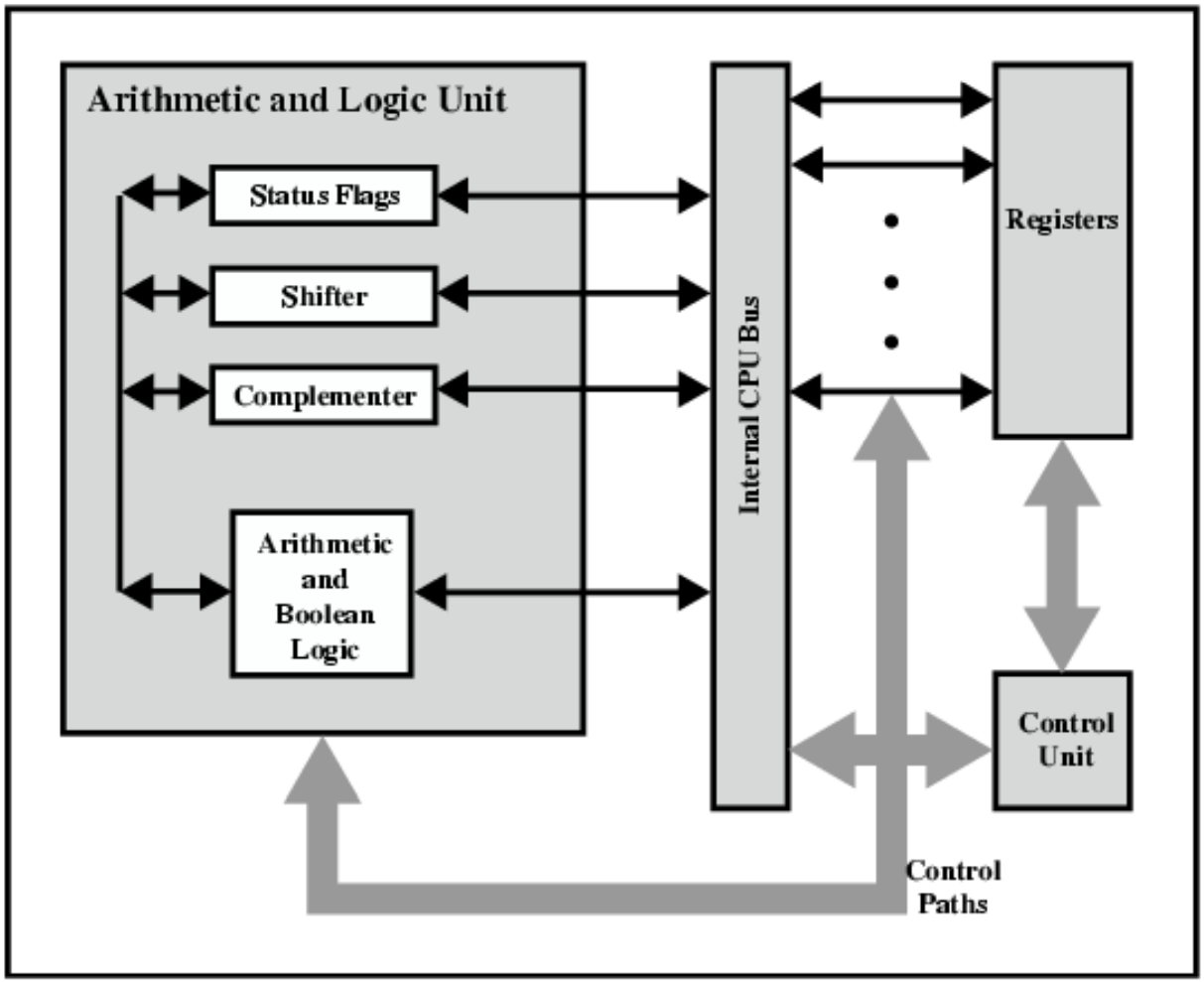
1. CPU Organization

CPU with system buses



1. CPU Organization

CPU internal structure



1. CPU Organization

To understand the organization of the CPU, let us consider the things that it must do:

- Fetch instruction: The processor reads an instruction from memory
- Interpret instruction: The instruction is decoded to determine what action is required
- Fetch data: The execution of an instruction may require reading data from memory or an I/O module
- Process data: The execution of an instruction may require performing some arithmetic or logical operation on data
- Write data: The results of an execution may require writing data to memory or an I/O module

1. CPU Organization

To do the basic operations, it should be clear that the processor needs to:

- Store some data temporarily
- Remember the location of the last instruction so that it can know where to get the next instruction.
- Store instructions and data temporarily while an instruction is being executed

In other words, the CPU needs a small internal memory

2. Registers Organization

CPU must have some working space (temporary storage) called registers

- High speed storage locations
- Number and function vary between processor designs
- One of the major design decisions
- Top level of memory hierarchy

2. Registers Organization

The registers in the processor perform two roles:

i. User-visible registers: Enable the CPU to minimize main memory references by optimizing use of registers

ii. Control and status registers: Used by the control unit to control the operation of the processor and by privileged, operating system programs to control the execution of programs.

There is not a clean separation of registers into these two categories. For example, on x86, the program counter is user visible.

2.1. User Visible Registers

A user-visible register is one that may be referenced by means of the machine language that the processor executes. We can characterize these in the following categories:

- General Purpose
- Data
- Address
- Condition Codes

2.1. User Visible Registers

General-purpose registers can be assigned to a variety of functions by the programmer

- That is, any general-purpose register can contain the operand for any opcode. This provides true general-purpose register use
  - Often, however, there are restrictions. For example, there may be dedicated registers for stack operations. In some cases, general-purpose registers can be used for addressing functions. In other cases, there is a partial or clean separation between data registers and address registers.

2.1. User Visible Registers

Data registers: may be used only to hold data and cannot be employed in the calculation of an operand address

Address registers: may themselves be somewhat general purpose, or they may be devoted to a particular addressing mode. Examples include the following:

- Segment pointers
- Index registers
- Stack pointer

2.1. User Visible Registers

condition codes (also referred to as flags) are bits set by the processor hardware as the result of operations

- For example, an arithmetic operation may produce a positive, negative, zero, or overflow result.
- Codes may be tested (read) by programs as part of a conditional branch operation (JZ inst.)
- Codes can not be set by programs

2.1. User Visible Registers

How many?

- From 8 to 32 registers
- Fewer = more memory references

How big?

- Large enough to hold full address
- Large enough to hold full word
- Often possible to combine two data registers

2.2. Control & Status Registers

Used by the control unit to control the operation of the processor and the execution of programs

- Used during the fetching, decoding and execution of instructions
- Many are not visible to the programmer, some are visible but can not be (easily) modified

2.2. Control & Status Registers

Typical control and status registers include:

- PC: Contains the address of an instruction to be fetched
- IR: Contains the instruction most recently fetched
- MAR: Contains the address of a location in memory
- MBR: Contains a word of data to be written to memory or the word most recently read
- Program Status Word (PSW): typically contains condition codes plus other status information

2.2. Control & Status Registers

Program Status Word (PSW):

- Sign: sign bit of the result of the last arithmetic operation
- Zero: Set when the result is 0
- Carry: Set if an operation resulted in a carry (addition) into or borrow (subtraction) out of a high-order bit. Used for multiword arithmetic operations
- Equal: Set if a logical compare result is equality
- Overflow: Used to indicate arithmetic overflow
- Interrupt Enable/Disable: Used to enable or disable interrupts
- Supervisor: Indicates whether the processor is executing in supervisor or user mode. Certain privileged instructions can be executed only in supervisor mode, and certain areas of memory can be accessed only in supervisor mode

2.3. Example of CPU Register Organizations

Processor Register Organization is generally called the Programming Model

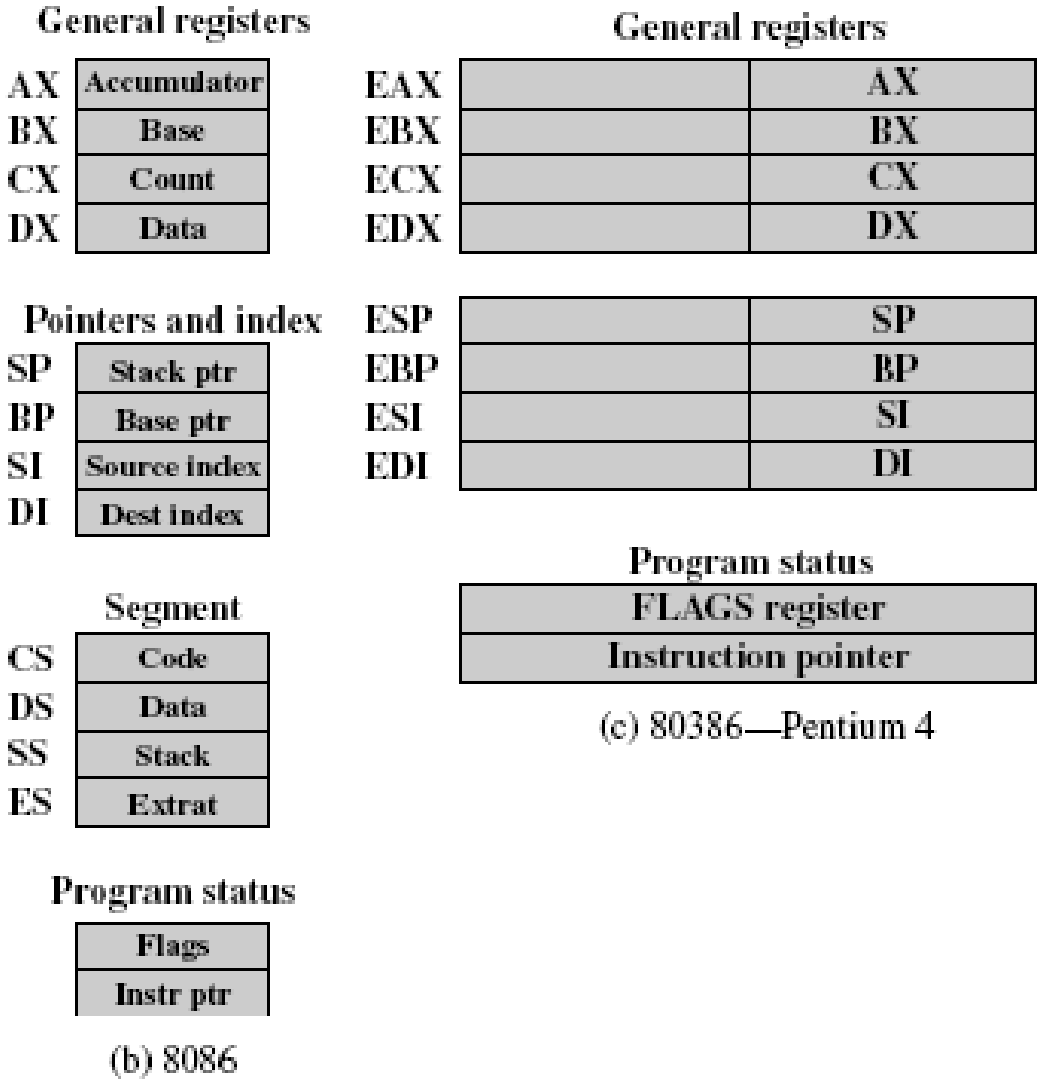
- it represents an internal register map for the programmer to write assembly programs

2.3. Example of CPU Register Organizations

The Intel 8086 is 16-bit processor with 20 bits address bus and 16 bit data bus. It has:

- Four 16-bit data registers (AX, BX, CX, and DX)
  - addressable on a byte or 16-bit basis
  - can be used as general purpose in some instructions
- Four 16-bit pointer and index reg. (SP, BP, SI and DI)
- Four 16-bit segment registers (CS, DS, SS, and ES)
  - Physical PC address = CS*16 + PC
- Instruction (or program) pointer register
- 16-bit flags register with 9 active flags

2.3. Example of CPU Register Organizations



2.3. Example of CPU Register Organizations

The Motorola MC68000 has 32-bit registers:

- Eight 32-bit data registers (D0-D7) that are used primarily for data manipulation and addressing as index registers. The width of the registers allows 8,16, and 32-bit data operations, determined by opcode.
- Nine 32-bit address registers (A0-A7)
  - Two A7 registers used as stack pointer (user or OS)
- 32-bit program counter
- 16-bit status register

2.3. Example of CPU Register Organizations

The Zilog Z80 has:

- Six 8-bit general purpose registers: B, C, D, E, H, and L that can be used to load or copy data
- 8-bit Accumulator A used to store 8-bit data and to perform ALU operations
- 8-bit flag register F that includes six flags that are set or reset according to the result
- Four 16-bit address registers: IX, IY, SP, and PC
- Two special purpose registers: I and R

2.3. Example of CPU Register Organizations

Zilog Z80 registers

Accumulator A	Flags F
B	C
D	E
H	L
Index Register IX	
Index Register IY	
Stack Pointer SP	
Program Counter PC	
Interupt Vector I	Memory Refresh R

2.3. Example of CPU Register Organizations

Example x86 Program to calculate the sum of two numbers:

- MOV EAX, 20H ; EAX = 20H (MOV = move)
- MOV EBX, 3AH ; EBX = 3AH
- ADD EAX, EBX ; EAX = EAX + EBX

Example Z80 Program to calculate the sum of two numbers:

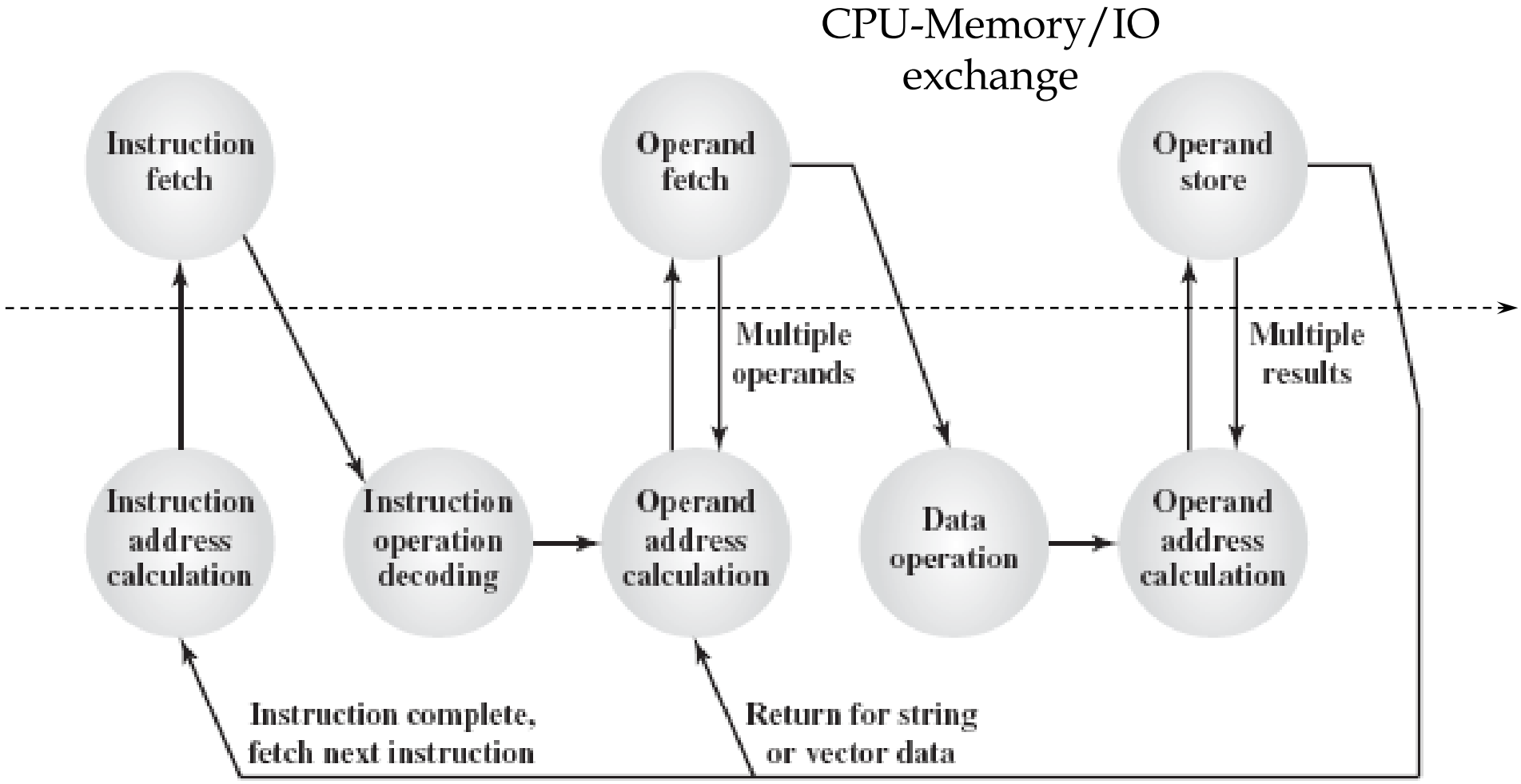
- LD A, 20H ; Accumulator A = 20H (LD = load)
- LD B, 3AH ; General purpose register B = 3AH
- ADD A, B ; A = A + B

3. Instruction Cycle

The basic direct instruction cycle contains the following states:

- Instruction address calculation (IAC)
- Instruction fetch (IF)
- Instruction operation decoding (IOD)
- Operand address calculation (OAC)
- Operand fetch (OF)
- Data operation (DO)
- Operand store (OS)

3. Instruction Cycle



Instruction Cycle State Diagram (Direct Cycle)

Source: Computer Organization and Structure, by W. Stallings

3. Instruction Cycle

For any given instruction cycle, some states may be null and others may be visited more than once

Examples:

- LOAD A,30H instruction involves the sequence: IAC, IF, IOD, OAC, and OF
- ADD A,30H instruction involves the sequence: IAC, IF, IOD, and DO
- JUMP 200H instruction involves IAC, IF, and IOD

3. Instruction Cycle

Indirect Cycle (mode)

- The instruction operand is an address
- The memory fetch get the address
- Additional memory access is required to fetch the data (multiple OF state)

4. Interrupts

Definition

Interrupt is a process by which other modules (e.g. I/O) may interrupt normal sequence of processing

- The process is asynchronous

Interrupts improve processing efficiency

- Interrupts allow the CPU to respond to peripherals on demands while it is free to perform other tasks
- In the polling method, the CPU remains in a loop continuously checking the peripheral

4. Interrupts

Classes of interrupts:

- Program
  - e.g. overflow, division by zero
- Timer
  - Generated by internal processor timer
  - To perform certain functions on regular basis
- I/O
  - from I/O devices
- Hardware failure
  - e.g. memory parity error

4. Interrupts

Types of Interrupts

- Hardware interrupts: Interrupt generated by Hardware
- Software Interrupts: Interrupt generated by Software
- Maskable Interrupts: Interrupt can be enabled/disabled or delayed.
- Nonmaskable Interrupts: Interrupt always enabled, can not be delayed.
- Vectored Interrupts: The address of the Interrupt Service Routine (ISR) is hard wired. The ISR is the software routine (program) invoked to respond to an interrupt
- Non-vectored Interrupts: The address of the ISR needs to be supplied

4.1. Interrupt Cycle

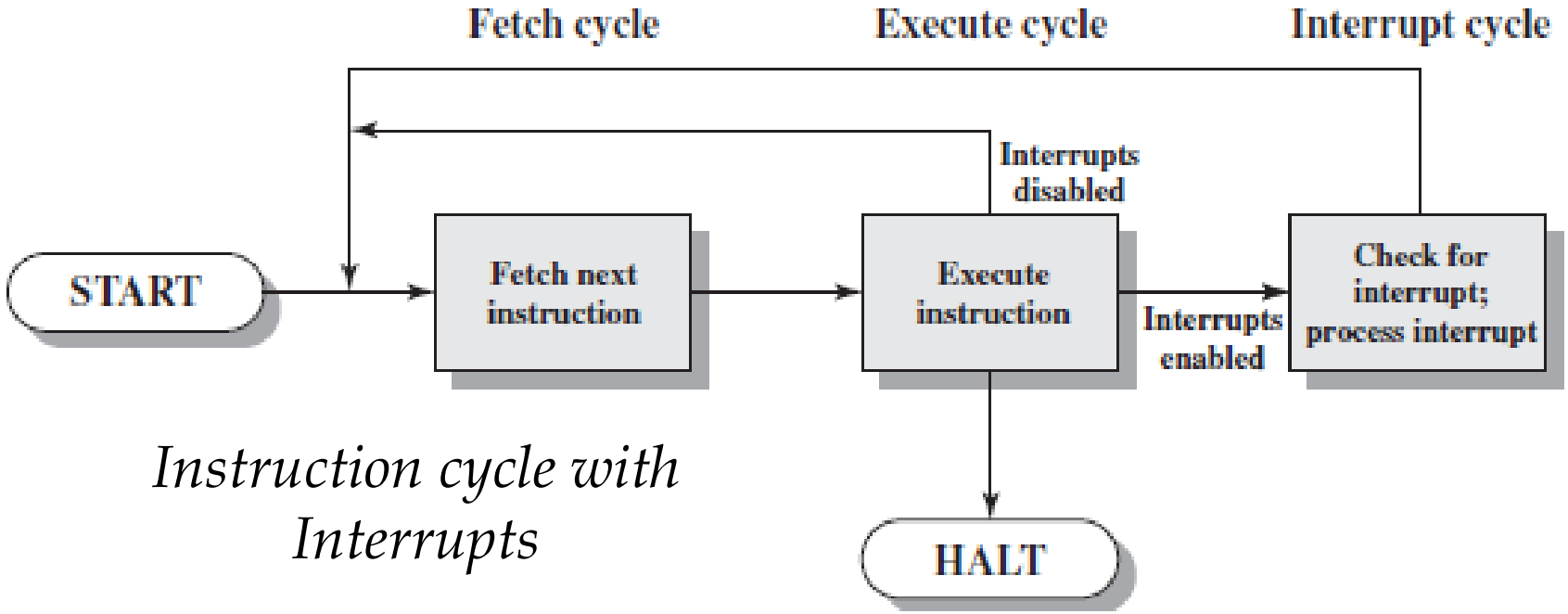
Interrupt Cycle

- CPU checks for interrupt at the end of the current instruction
- When CPU receives an interrupt signal, it suspends the currently executed program and jumps to an ISR to respond to the incoming interrupt
- The interrupt is ignored if it is maskable or disabled
- The address of the ISR can be:
  - Already known by the CPU (vectored Interrupt)
  - Supplied by the peripheral to the CPU (Non-vectored Interrupt)

4.1. Interrupt Cycle

Program execution

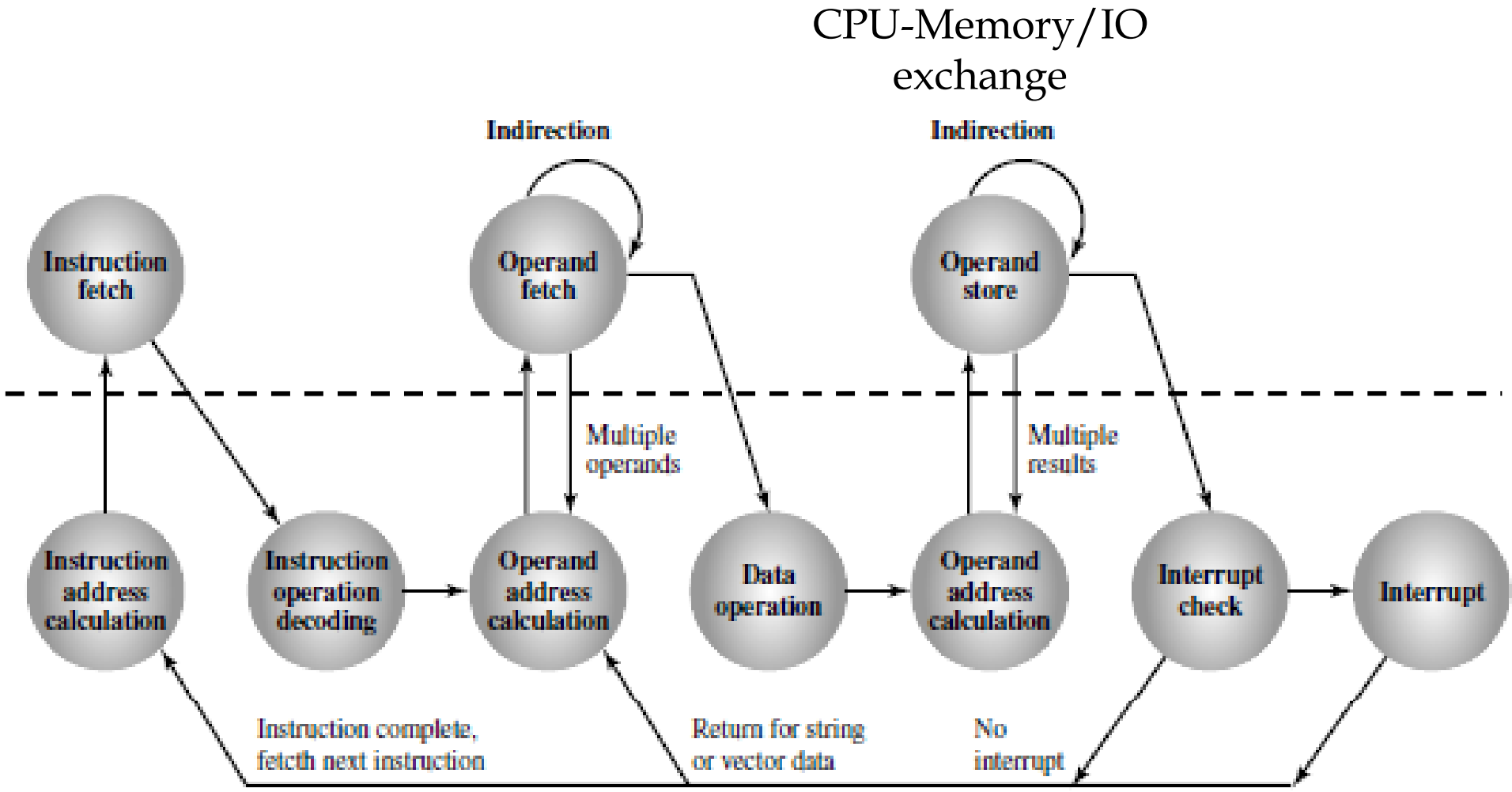
- It consists of repeating Fetch, Execute and Interrupt cycles



Instruction cycle with Interrupts

Source: Computer Organization and Structure, by W. Stallings

4.1. Interrupt Cycle



Instruction Cycle State Diagram with Interrupt cycle

Source: Computer Organization and Structure, by W. Stallings

4.1. Interrupt Cycle

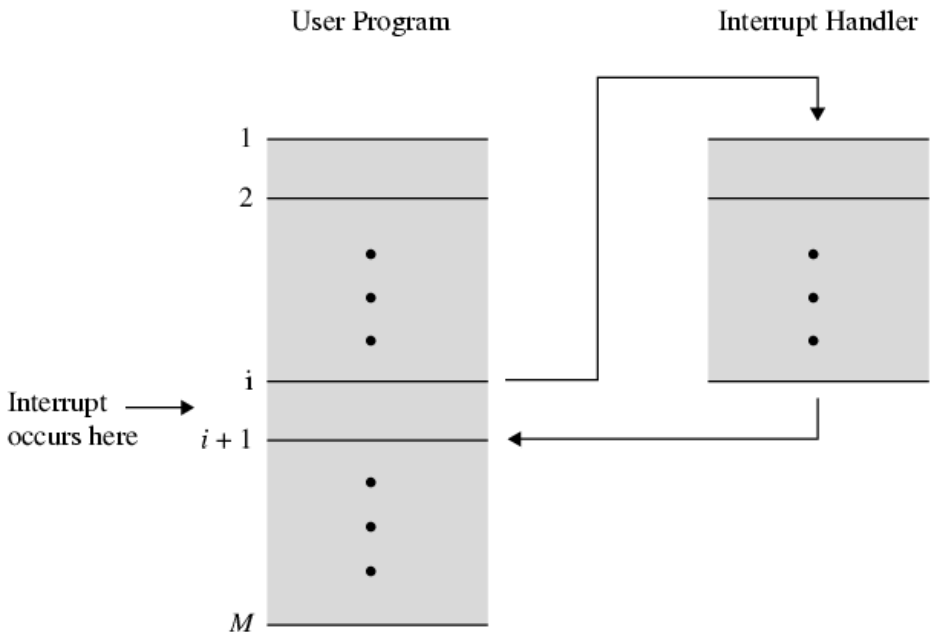
If interrupt pending:

- The CPU suspends execution of current program
- Save context
- Set PC to start address of ISR
- Process interrupt
- Restore context and continue current program

4.1. Interrupt Cycle

User program does not have to contain any special code to accommodate interrupts

CPU and OS are responsible for suspending the user program and then resuming it



Transfer of Control via Interrupt

4.2. Multiple Interrupts

We discussed only on the occurrence of a single interrupt, however, multiple interrupts can occur

Two approaches can be taken to dealing with multiple interrupts:

I. Disable interrupts
II. Define priorities

4.2. Multiple Interrupts

Disable interrupts

- CPU will ignore further interrupts while processing one interrupt
- Interrupts remain pending and are checked after current interrupt has been processed
- Interrupts handled in sequence as they occur

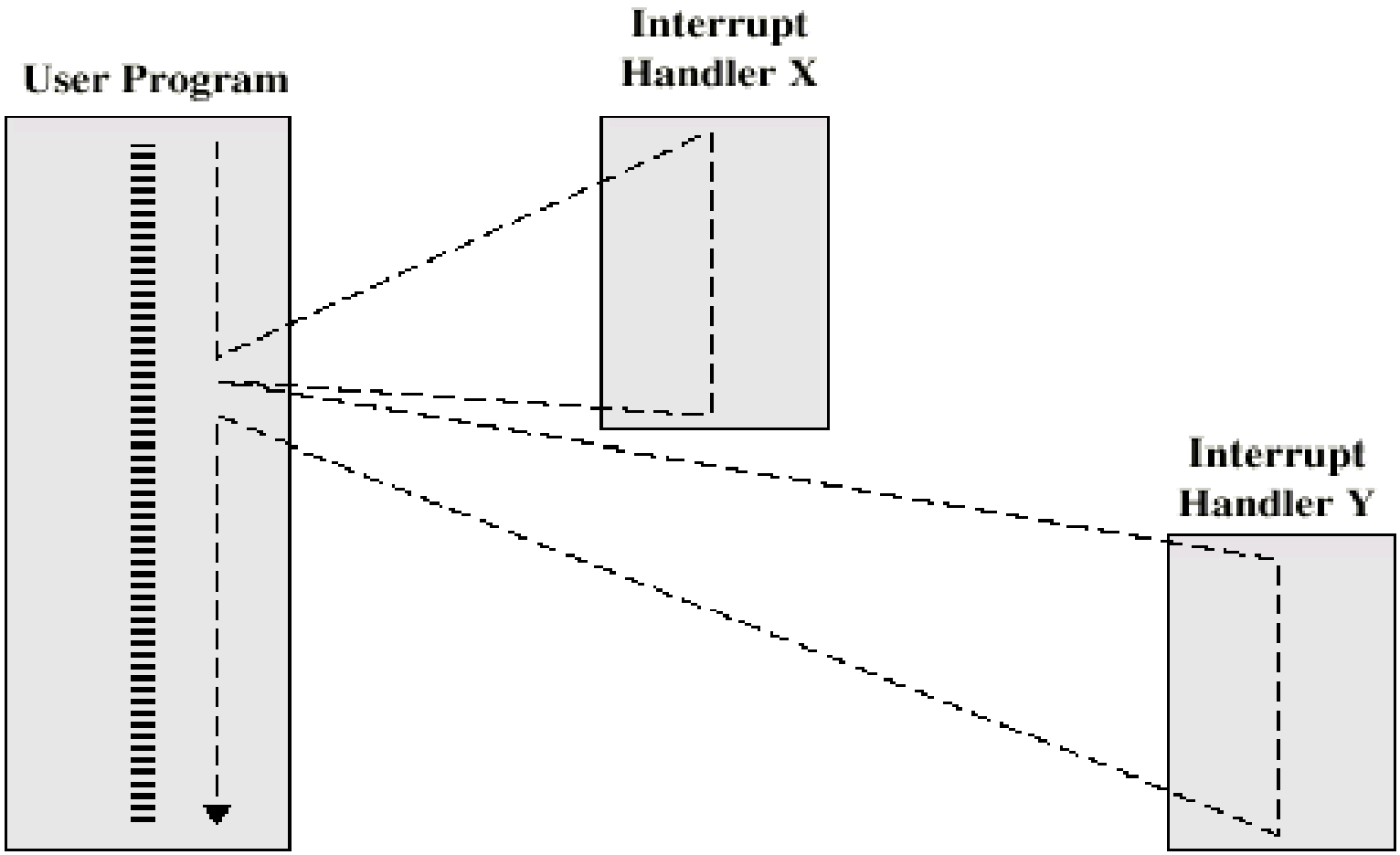
Advantage

- Simple approach

Drawback

- It does not take into account relative priority or time-critical needs

4.2. Multiple Interrupts



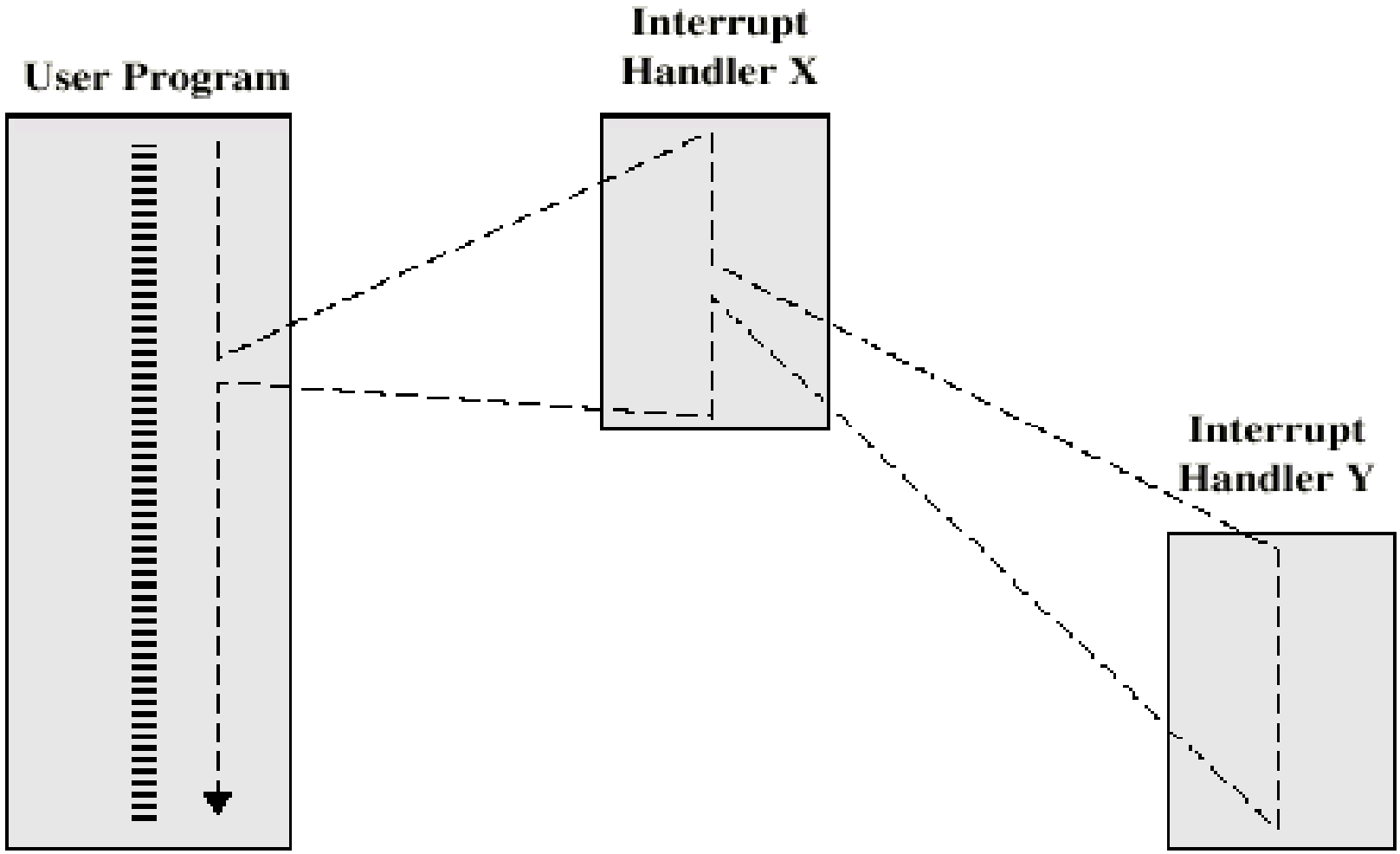
Sequential Interrupts

4.2. Multiple Interrupts

Define priorities

- Low priority interrupts can be interrupted by higher priority interrupts
- When higher priority interrupt has been processed, CPU returns to previous interrupt

4.2. Multiple Interrupts



Nested Interrupts

4.2. Multiple Interrupts

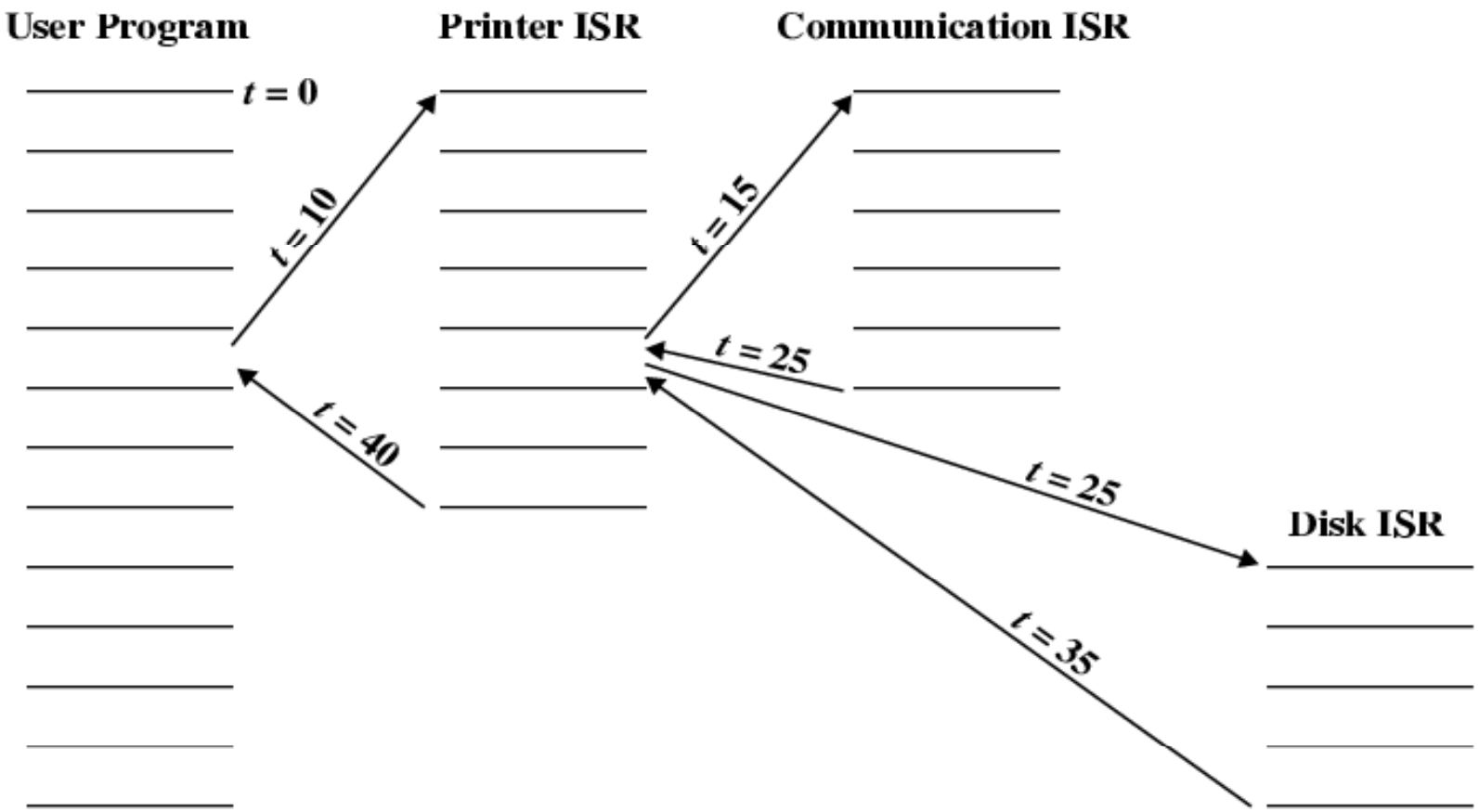
Example:

Consider a system with 3 I/O devices: a printer, a disk, and a communications line, with increasing priorities of 2,4, and 5, respectively

- t=0, user program begins
- t=10, a printer interrupt occurs
- t=15, a communications interrupt occurs
- t=20, a disk interrupt occurs

Give time sequence of the interrupts (assume all ISRs take 10 units)

4.2. Multiple Interrupts



Example of Time Sequence of Multiple Interrupts

5. CPU Examples

In this section we examine some of the details of some processor organizations, concentrating on common elements in single processors:

- Z80 processor
- ARM processor
- X86 processor family

5. CPU Examples

Z80 Processor: Programming model

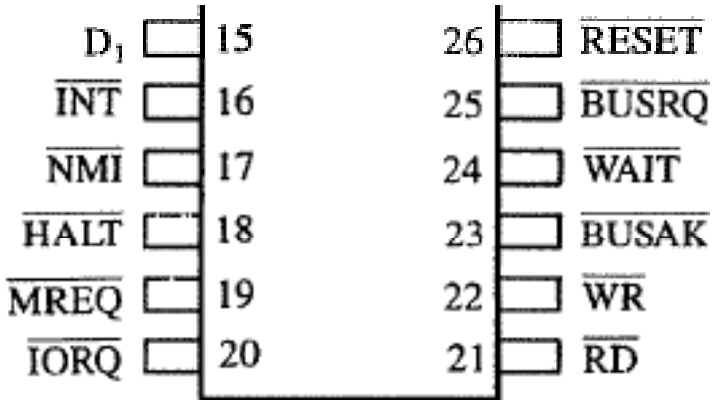
- 8-bit processor
- 8-bit data bus and 16-bit address bus
- 8x8-bit registers and 6x16-bit address registers
- 158 instructions, from 1-byte to 4-bytes instructions
- 256 input ports
- 256 output ports

5. CPU Examples

Z80 Processor: Instruction Cycle

- Z80 instruction cycle consists of one to six machine cycles
- Machine cycle can be opcode fetch, memory read and write, and I/O read and write
- Each machine cycle consists of three to six T-states (clock cycles) in duration
  - Opcode fetch: 4 clock cycles
  - Memory read and write: 3 clock cycles
  - I/O read and write: 4 clock cycles

5. CPU Examples



Z80 Processor: Interrupts

- Reset signal: /RESET
  - Z80 starts executing ISR at 0000H
- 1 Non-maskable interrupt: /NMI
  - Z80 starts executing ISR at address 0066H
- 1 maskable interrupt: /INT
  - Can be enabled and disabled using EI and DI inst
  - ISR address depends on interrupt mode
- 8 software interrupts RST 0 to RST 7

Z80 Processor: Interrupts

- Z80 saves the context and PC in the stack
- Z80 fetch the first instruction of ISR. For maskable interrupt (/INT), the address of ISR depends on the interrupt mode
  - Z80 has three processing modes for maskable interrupts that can be set using IM0, IM1, and IM2 instruction
- After executing the ISR, Z80 restores PC from stack and continue execution of the program

5. CPU Examples

Z80 Processor: Interrupt modes

- IM0: Z80 reads a byte from the data bus and executes the ISR at one of the eight locations 0000H to 0038H (data*8)
- IM1: Z80 starts executing ISR at address 0038H
- IM2: Z80 the program execution is transferred to the memory location: (I register * 256) + Data bus value
  - I register store high order 8 bits of the 16 bits ISR address
  - The low 8 bits must be supplied by the I/O device

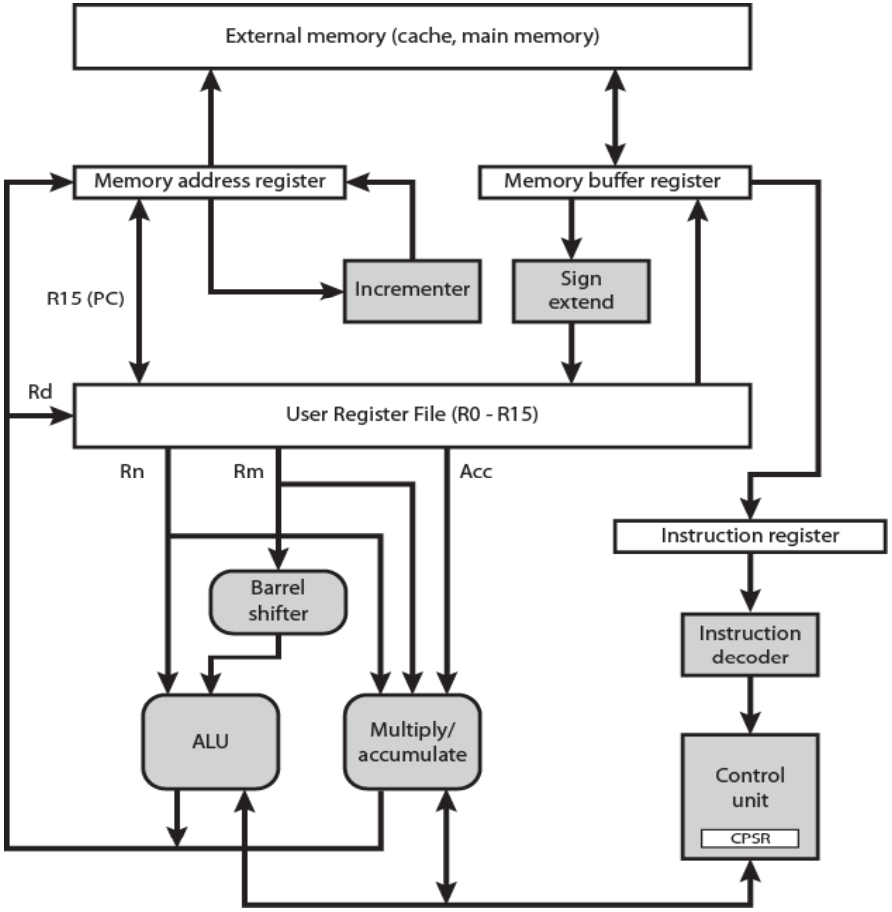
5. CPU Examples

ARM Processor

- ARM (Advanced RISC Machine) processor is the most used IP (Intellectual Property) CPU
  - IP is a functional block that can be licensed and incorporated as building block within ASIC chip or FPGA design
- 15 billions ARM-based chip are sold in 2014 (400 millions processors sold by Intel)

5. CPU Examples

ARM Processor Organization



5. CPU Examples

ARM Processor Organization

- Many variations depending on ARM version
- A load/store model of data processing, in which operations only perform on operands in registers and not directly in memory
- Data goes to register file (Set of 32 bit registers)
- Rotation and shift unit (before ALU)
- Conditional execution of instructions minimizes the need for conditional branch instructions
- Multiply/accumulate unit

5. CPU Examples

ARM Register Organization

- 37 x 32-bit registers
  - R0-R12 and R15(PC)
  - 6 x R13(SP)
  - 6 x R14 (LR)
  - R8-R12 for Fast Interrupt
  - 6 (1 for each mode) current program status registers (CPSR)
- 6 saved program status register (SPSR)

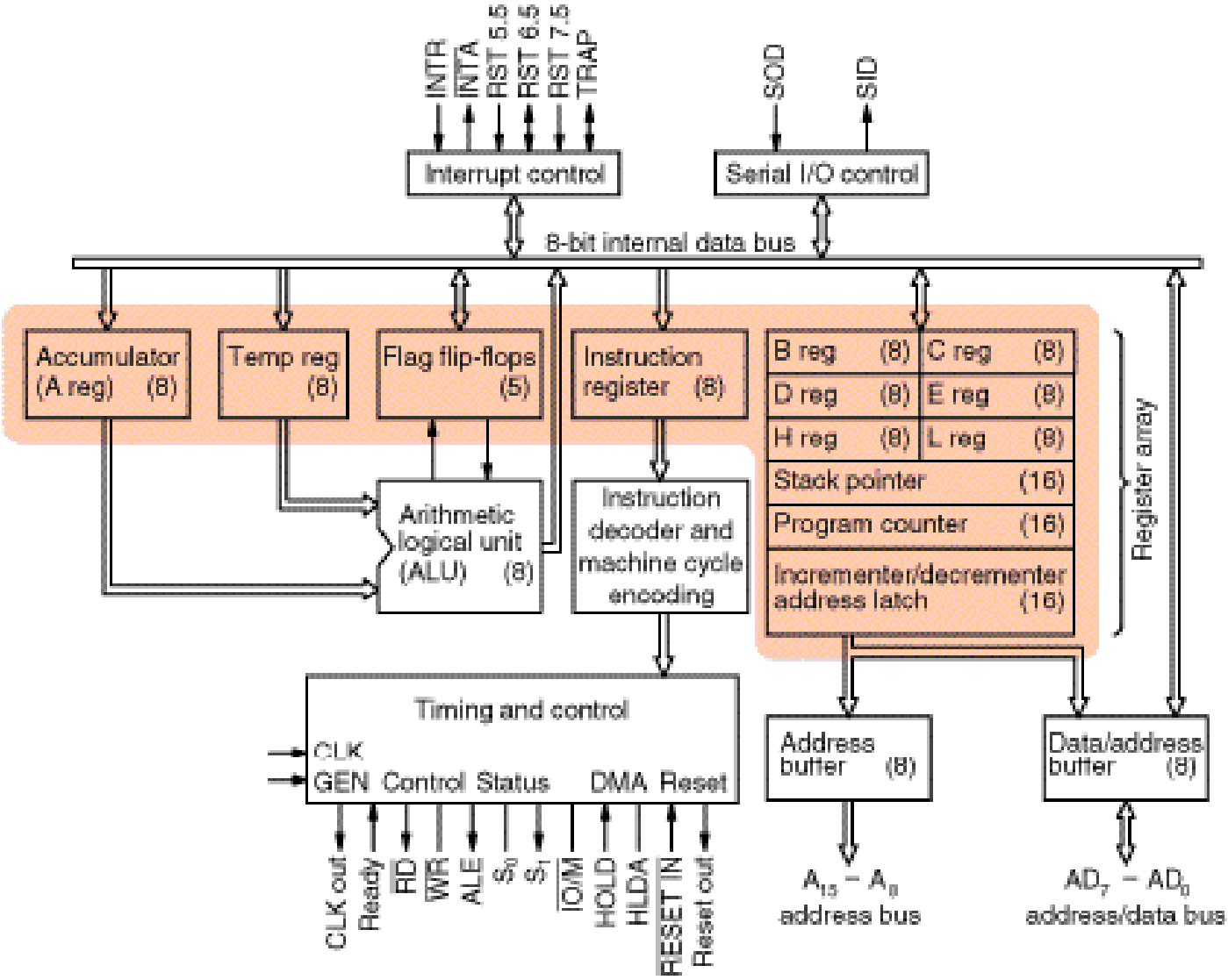
5. CPU Examples

ARM Register Organization

- R13 normally stack pointer (SP)
  - Each exception mode has its own R13
- R14 link register (LR)
  - Subroutine and exception mode return address
- R15 program counter

5. CPU Examples

Intel 8085



5. CPU Examples

X86 Family: Pentium 4 registers

- 8x32-bit GP user registers
- 6x16-bit segment registers
- One 32-bit flags register
- One 32-bit instruction counter
- 8x80-bit registers for floating point numbers
- Other control and status registers for floating point unit

5. CPU Examples

X86 Family: 80486 Instruction Cycle

80486 instruction cycle is divided to the following machine cycles:

- Fetch
- Decode stage 1
- Decode stage 2
- Execute
- Writeback

5. CPU Examples

X86 Family: Pentium Interrupt Processing

- Interrupts
  - Maskable
  - Nonmaskable
- Exceptions
  - Processor detected
  - Programmed
- Interrupt vector table
  - Each interrupt type assigned a number
  - Index to vector table
  - 256 * 32 bit interrupt vectors
- 5 priority classes

Summary

CPU Component and function

Registers organization

Instruction Cycle

Interrupts