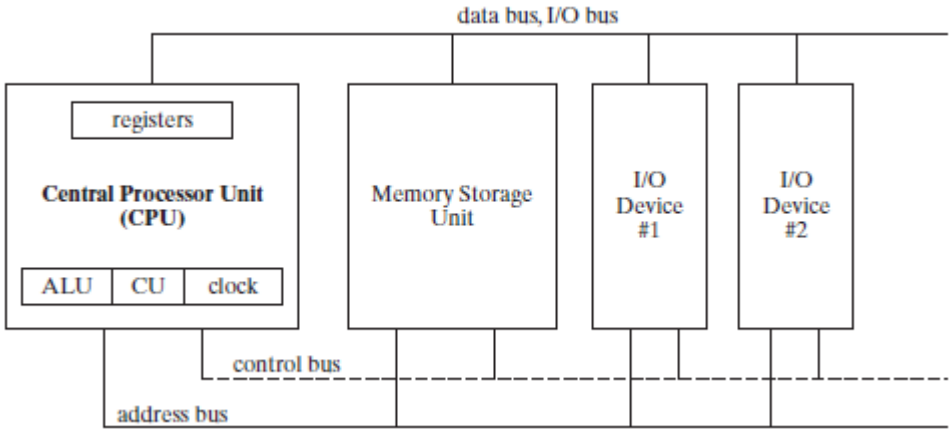
Created By: Eng. Ahmed M. Ayash
Modified and Presented by: ***Eng. Eihab S. El-Radie***

CHAPTER 2

IA-32 Processor Architecture

➢ Basic Microcomputer Design

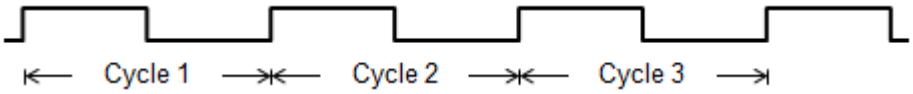


- The central processor unit (**CPU**) is where all the calculations and logic operations take place.
- The central processor unit (CPU) contains:
  - **Registers** a limited number of storage locations.
  - **Clock** *synchronizes* CPU operations.
  - Control unit (**CU**) coordinates sequence of execution steps.
  - Arithmetic logic unit (**ALU**) performs arithmetic operations such as addition and subtraction and logical operations such as AND, OR, and NOT.

- The **memory storage unit** is where instructions and data are held while a computer program is running.

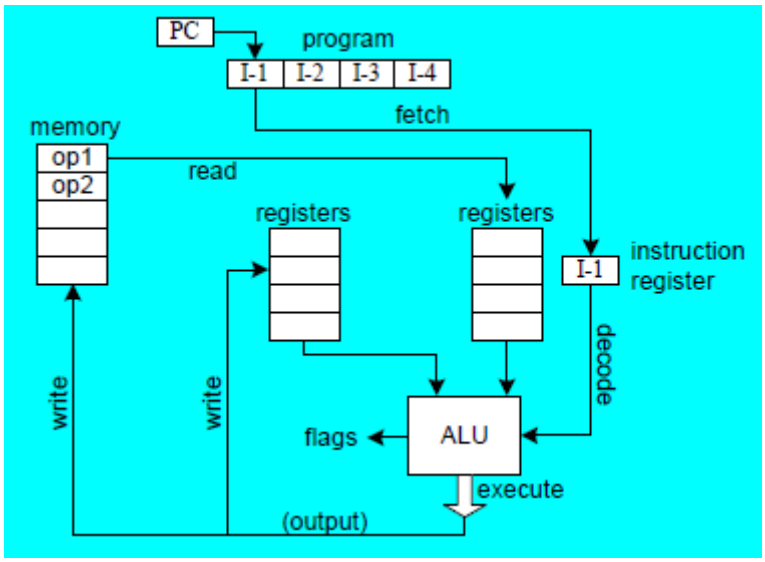
- **A bus** is a group of parallel wires that transfer data from one part of the computer to another.
- A computer system usually contains **four** bus types: data, I/O, control, and address.
  - **The data bus** transfers instructions and data between the CPU and memory.
  - **The I/O bus** transfers data between the CPU and the system input/output devices.
  - **The control bus** uses binary signals to synchronize actions of all devices attached to the system bus.
  - **The address bus** holds the addresses of instructions and data when the currently executing instruction transfers data between the CPU and memory.

- **Clock** Synchronizes Processor and Bus operations.



  - Clock cycle = Clock period = 1 / Clock rate.
  - Clock rate = Clock frequency = Cycles per second.

➢ Instruction Execution Cycle



- The execution of a single machine instruction can be divided into a sequence of individual operations. Three primary operations: **fetch**, **decode** and **execute**. Two more steps are required when the instruction uses a memory operand: **fetch operand** and **store output** operand.
  - **Fetch:** The control unit fetches the next instruction from the instruction queue and increments the instruction pointer (IP). The IP is also known as the program counter

  - **Decode:** The control unit decodes the instruction's function to determine what the instruction will do.

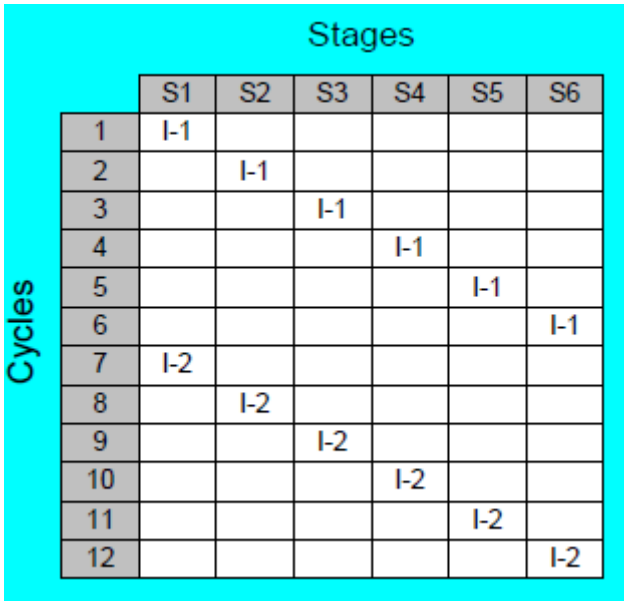
- **Fetch operands:** If the instruction uses an input operand located in memory, the control unit uses a read operation to retrieve the operand and copy it into internal registers. Internal registers are not visible to user programs.

- **Execute:** The ALU executes the instruction using the named registers and internal registers as operands and sends the output to named registers and/or memory. The ALU updates status flags providing information about the processor state.

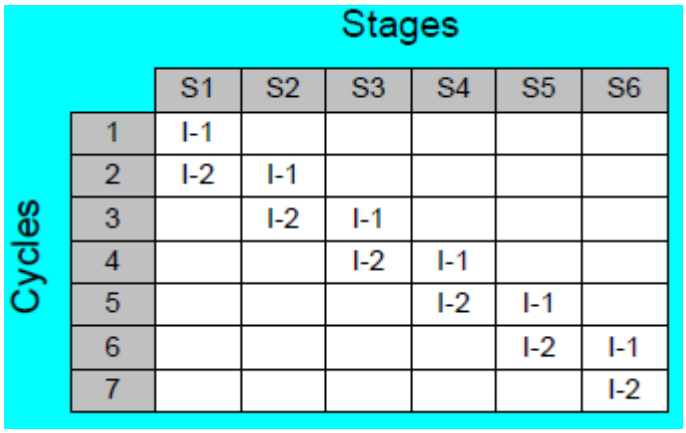
- **Store output operand:** If the output operand is in memory, the control unit uses a write operation to store the data.

➢ Multi-Stage Pipeline

- Pipelining makes it possible for a processor to execute instructions in parallel
- Instruction execution divided into discrete stages



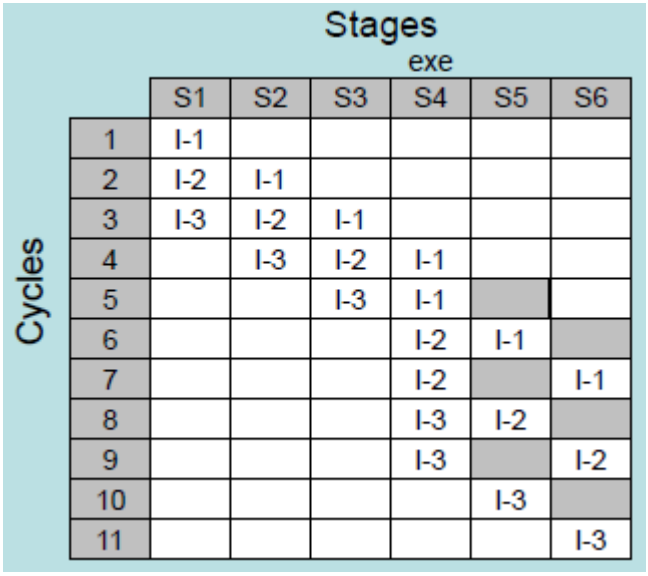
Example of a nonpipelined processor.



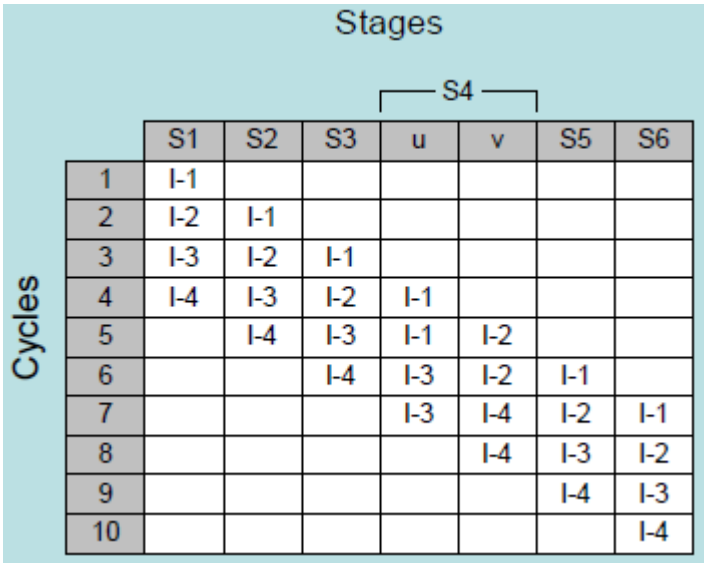
Example of a pipelined processor.

- **Nonpipelined Execution**
  - Many wasted cycles.
  - For *k* states and *n* instructions, the number of required cycles is: ***n*k***

- **Pipelined Execution**
  - For *k* states and *n* instructions, the number of required cycles is: ***k* + (*n* – 1)**



Example of Wasted Cycles (pipelined)

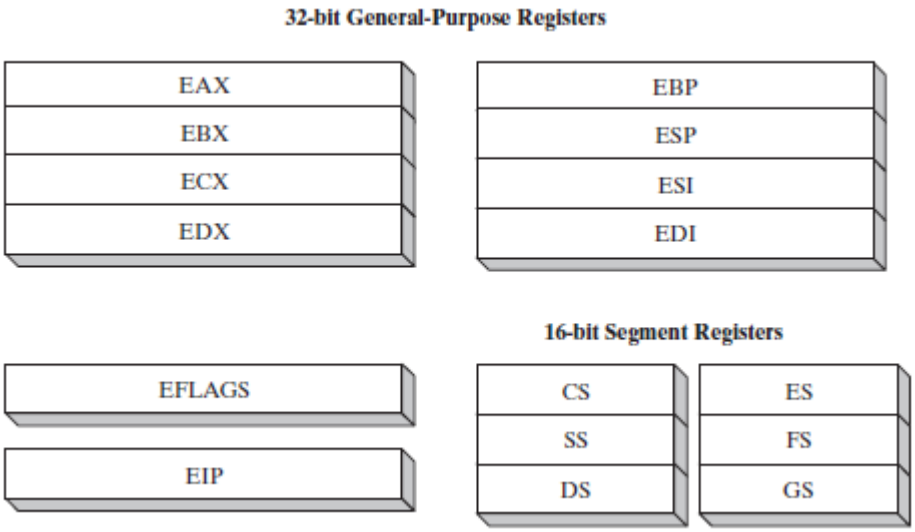


Superscalar Architecture

- **Wasted Cycles (pipelined)**
  - When one of the stages requires two or more clock cycles to complete, clock cycles are again wasted.
  - Assume that S4 requires 2 clock cycles to complete. As more instructions enter the pipeline, wasted cycles occur.
  - For k stages, where one stage requires 2 cycles, n instructions require: ***k + (2n – 1)*** cycles

- **A superscalar**
  - A superscalar processor has multiple execution pipelines, the processor has two execution pipelines : Called U and V pipes
  - Assume that stage S4 has 2 pipelines. Each pipeline still requires 2 cycles.
  - Second pipeline eliminates wasted cycles
  - For *k* stages and *n* instructions, number of cycles = ***k + n***

➢ Basic Program Execution Registers



- **Status Flags**
  - **Carry Flag:** Set when unsigned arithmetic result is out of range.
  - **Overflow Flag:** Set when signed arithmetic result is out of range.
  - **Sign Flag:** Copy of sign bit, set when result is negative.
  - **Zero Flag:** Set when result is zero.
  - **Auxiliary Carry Flag:** Set when there is a carry from bit 3 to bit 4.
  - **Parity Flag:** is set if the least-significant byte in the result contains an even number of 1 bits.

➢ Modes of Operation

- **Real-address mode**
  - Only 1 MByte of memory can be addressed
  - 20 bit addresses from hexadecimal 00000 to FFFFF.
  - The processor can run only one program at a time
  - Application programs are permitted to access any memory location.
  - The MS-DOS operating system runs in real-address mode, and Windows 95 and 98 can be booted into this mode.

- **Protected mode**
  - Each program can address a maximum of 4 GB of memory
  - 32 bit addresses (From 00000000 to FFFFFFFFh)
  - The processor can run multiple programs at the same time.
  - The operating system assigns memory to each running program
  - Programs are prevented from accessing each other's memory
  - MS-Windows and Linux run in protected mode.

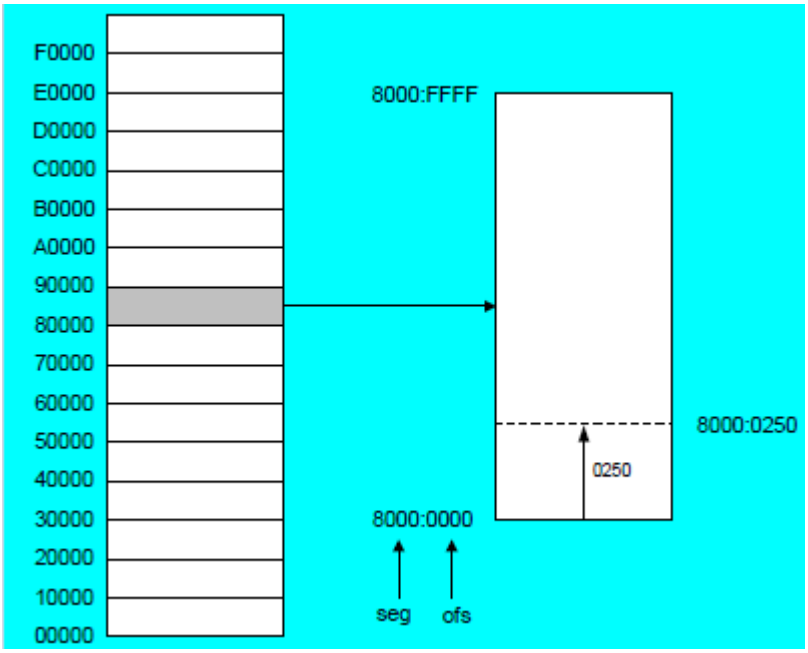
- **Virtual-8086 mode**
  - The computer runs in protected mode and creates a virtual 8086 machine with its own 1-MByte address space that simulates an 80x86 computer running in real address mode.
  - Windows NT, 2000, and XP, for example, create a virtual 8086 machine when you open a Command window. You can run many such windows at the same time, and each is protected from the actions of the others.

➢ Segmented memory

The original 8086 processor had only 16-bit registers, which cannot directly represent a 20-bit address.

- Came up with a scheme known as **segmented memory**. All of memory is divided into 64-kilobyte, (64-KByte) units called **segments**.

- Segmented memory addressing: absolute (linear) address is a combination of a 16-bit segment value added to a 16-bit offset.



Real-Address Mode, Segmented Memory Map

Each segment begins at an address having a zero for its last hexadecimal digit. Because the last digit is always zero, it is omitted when representing segment values.

20-Bit Linear Address Calculation

In real-address mode, the linear (or absolute) address is 20 bits, ranging from 0 to FFFFF hexadecimal. Programs cannot use linear addresses directly, so addresses are expressed using **two** 16-bit integers.
A ***segment-offset*** address includes the following:
• A 16-bit **segment** value, placed in one of the segment registers (CS, DS, ES, SS)
• A 16-bit **offset** value

- The CPU automatically converts a segment-offset address to a 20-bit linear address. Suppose a variable's hexadecimal segment-offset address is **08F1:0100**.

  - The CPU multiplies the segment value by 16 (10 hexadecimal) and adds the product to the variable's offset:

```
08F1h × 10h = 08F10h          (adjusted segment value)
Adjusted Segment value:       0  8  F  1  0
Add the offset:                  0  1  0  0
Linear address:               0  9  0  1  0
```

  - **Linear address = Segment × 10 (hex) + Offset**

Homework:

1. From Book (7th edition)

Section Review 2.1.5: 1, 2, 3, 4, 5

Section Review 2.2.5: 1,2,3,4

2. Others

1. What is the duration of a single clock cycle in a 3-GHz processor?

2. Define pipelined execution.

3. In a 5 -stage non-pipelined processor, how many clock cycles would it take to execute 2 instructions?

4. In a 5-stage single-pipelined processor, how many clock cycles would it take to execute 8 instructions?

5. What is the range of addressable memory in protected mode?

6. What is the range of addressable memory in real-address mode?

7. In Real -address mode. Convert the following segment-offset address to a linear address: 0950:0100

Quiz Next Week in Chapter2