Chapter 7
CPU Architecture

Outline

- Introduction to 8085 microprocessor
- 8085 hardware model
- Programmable registers
- Instruction format
- Addressing modes
- Addressing format
- Instruction set
- Instruction execution
- Fetch and execution cycle
- Microprogramming concept

What is a Microprocessor?

- **Microcomputer**
- The term microcomputer is generally synonymous with personal computer, or a computer that depends on a microprocessor.
  - Microcomputers are designed to be used by individuals, whether in the form of PCs, workstations or notebook computers.
  - A microcomputer contains a CPU on a microchip (the microprocessor), a memory system (typically ROM and RAM), a bus system and I/O ports, typically housed in a motherboard.

What is a Microprocessor?

- **Microprocessor:**
- A silicon chip that contains a CPU. In the world of personal computers, the terms microprocessor and CPU are used interchangeably.
  - A microprocessor (sometimes abbreviated μP) is a digital electronic component with miniaturized transistors on a single semiconductor integrated circuit (IC).
  - One or more microprocessors typically serve as a central processing unit (CPU) in a computer system or handheld device.
  - They made possible the advent of the microcomputer being At the heart of all personal computers and most working stations.
  - Microprocessors also control the logic of almost all digital devices, from clock radios to fuel-injection systems for automobiles.

What is a Microprocessor?

- The word comes from the combination micro and processor.
  - Processor means a device that processes data. In this context processor means a device that processes numbers, specifically binary numbers, 0's and 1's.
- Micro is a new addition.
  - In the late 1960's, processors were built using discrete elements. These devices performed the required operation, but were too large and too slow.
  - In the early 1970's the microchip was invented. All of the components that made up the processor were now placed on a single piece of silicon.
  - The size became several thousand times smaller and the speed became several hundred times faster. The "Micro"Processor was born.

Definition of the Microprocessor

The microprocessor is a programmable device that takes in numbers, performs on them arithmetic or logical operations according to the program stored in memory and then produces other numbers as a result.

- Lets expand each of the underlined words:
  - Programmable device: The microprocessor can perform different sets of operations on the data it receives depending on the sequence of instructions supplied in the given program.
  - By changing the program, the microprocessor manipulates the data in different ways.
  - Instructions: Each microprocessor is designed to execute a specific group of operations, called an instruction set. This instruction set defines what the microprocessor can and cannot do.

Definition (Contd.)

- Takes in: The data that the microprocessor manipulates must come from somewhere.
  - It comes from what is called “input devices”.
  - These are devices that bring data into the system from the outside world.
  - These represent devices such as a keyboard, a mouse, switches, and the like.
- Numbers: The microprocessor has a very narrow view on data. It only understands binary numbers.
  - A binary digit is called a bit (which comes from binary digit).
  - The microprocessor recognizes and processes a group of bits together. This group of bits is called a “word”.
  - The number of bits in a Microprocessor’s word, is a measure of its “abilities”.

Definition (Contd.)

- Words, Bytes, etc.
  - The earliest microprocessor (the Intel 8088 and Motorola's 6800) recognized 8-bit words.
    - They processed information 8-bits at a time. That's why they are called "8-bit processors". They can handle large numbers, but in order to process these numbers, they broke them into 8-bit pieces and processed each group of 8-bits separately.
  - Later microprocessors (8086 and 68000) were designed with 16-bit words.
    - A group of 8-bits were referred to as a "half-word" or "byte".
    - A group of 4 bits is called a "nibble".
    - Also, 32 bit groups were given the name "long word".
  - Today, all processors manipulate at least 32 bits at a time and there exists microprocessors that can process 64,128 bits or more at a time.

Definition (Contd.)

- Arithmetic and Logic Operations:
  - Every microprocessor has arithmetic operations such as add and subtract as part of its instruction set.
    - Most microprocessors will have operations such as multiply and divide.
    - Some of the newer ones will have complex operations such as square root.
  - In addition, microprocessors have logic operations as well. Such as AND, OR, XOR, shift left, shift right, etc.
  - Again, the number and types of operations define the microprocessor's instruction set and depends on the specific microprocessor.

Definition (Contd.)

- Program: A program is a sequence of instructions that bring data into the microprocessor, processes it and sends it out.
  - There are many programming languages (C, C++, FORTRAN, and JAVA…) However, these programming languages can be grouped into three main levels (these days a fourth level is developing).
- Programming Languages
  - Machine language
    - Machine language is the lowest level programming language. It is a language intended to be understood by the microprocessor (the machine) only.
      In this language, every instruction is described by binary patterns.

      e.g. 11001101 may mean 1 + 2

      This is the form in which instructions are stored in memory. This is the only form that the microprocessor understands.

Definition (Contd.)

- Programming Languages
  - Assembly language
    - This language is more understandable by humans. In this language, the binary patterns are assigned *mnemonics* (short abbreviated names).

      e.g. “Add 1,2” is assigned to the machine language pattern 11001101 mentioned above to refer to the operation 1+2.

      There is usually one assembly language instruction for each machine language instruction.

Definition (Contd.)

- Programming Languages
  - High level languages
    - These are languages like C, PASCAL and FORTRON. These are more natural for humans to use than assembly or machine languages. They are also more compact (i.e. it takes less statements to write the program).

      One high level instruction translates into many assembly or machine language instructions.

      e.g. x = y + z may translate into:

      MOV 1000, R_1
      MOV 1004, R_2
      ADD R_1, R_2
      MOV R_1, 1008

Definition (Contd.)

- Programming Languages
  - The new level being developed: is ultra high level languages which would contain things like C++, and JAVA.
    - Here a single instruction may translate into hundreds of assembly or machine language instructions.

Definition (Contd.)

- Stored in memory :
  - First, what is memory?
    - Memory is the location where information is kept while not in current use.
    - Memory is a collection of storage devices. Usually, each storage device holds one bit.
    - Also, in most kinds of memory, these storage devices are grouped into groups of 8.
    - These 8 storage locations can only be accessed together. So, one can only read or write in terms of bytes to and form memory.
    - Memory is usually measured by the number of bytes it can hold. It is measured in Kilos, Megas and lately Gigas.
    - A Kilo in computer language is 2^{10} =1024. So, a KB (KiloByte) is 1024 bytes. Mega is 1024 Kilos and Giga is 1024 Mega.

Definition (Contd.)

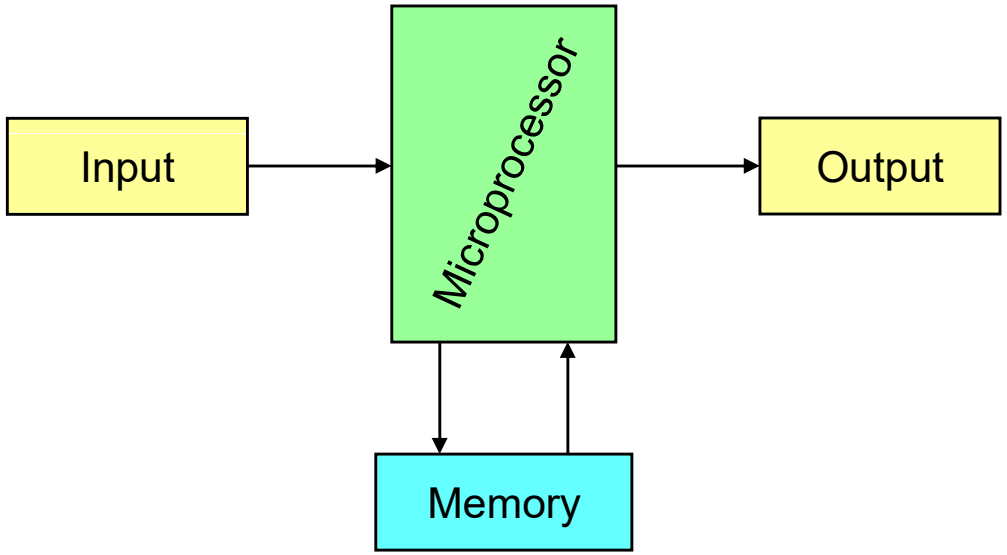
- Stored in memory:
  - When a program is entered into a computer, it is stored in memory. Then as the microprocessor starts to execute the instructions, it brings the instructions from memory one at a time.
  - Memory is also used to hold the data.
    - The microprocessor reads (brings in) the data from memory when it needs it and writes (stores) the results into memory when it is done.

Definition (Contd.)

- Produces: For the user to see the result of the execution of the program, the results must be presented in a human readable form.
  - The results must be presented on an output device.
  - This can be the monitor, a paper from the printer, a simple LED or many other forms.

A Microprocessor-based system

From the above description, we can draw the following block diagram to represent a microprocessor-based system:



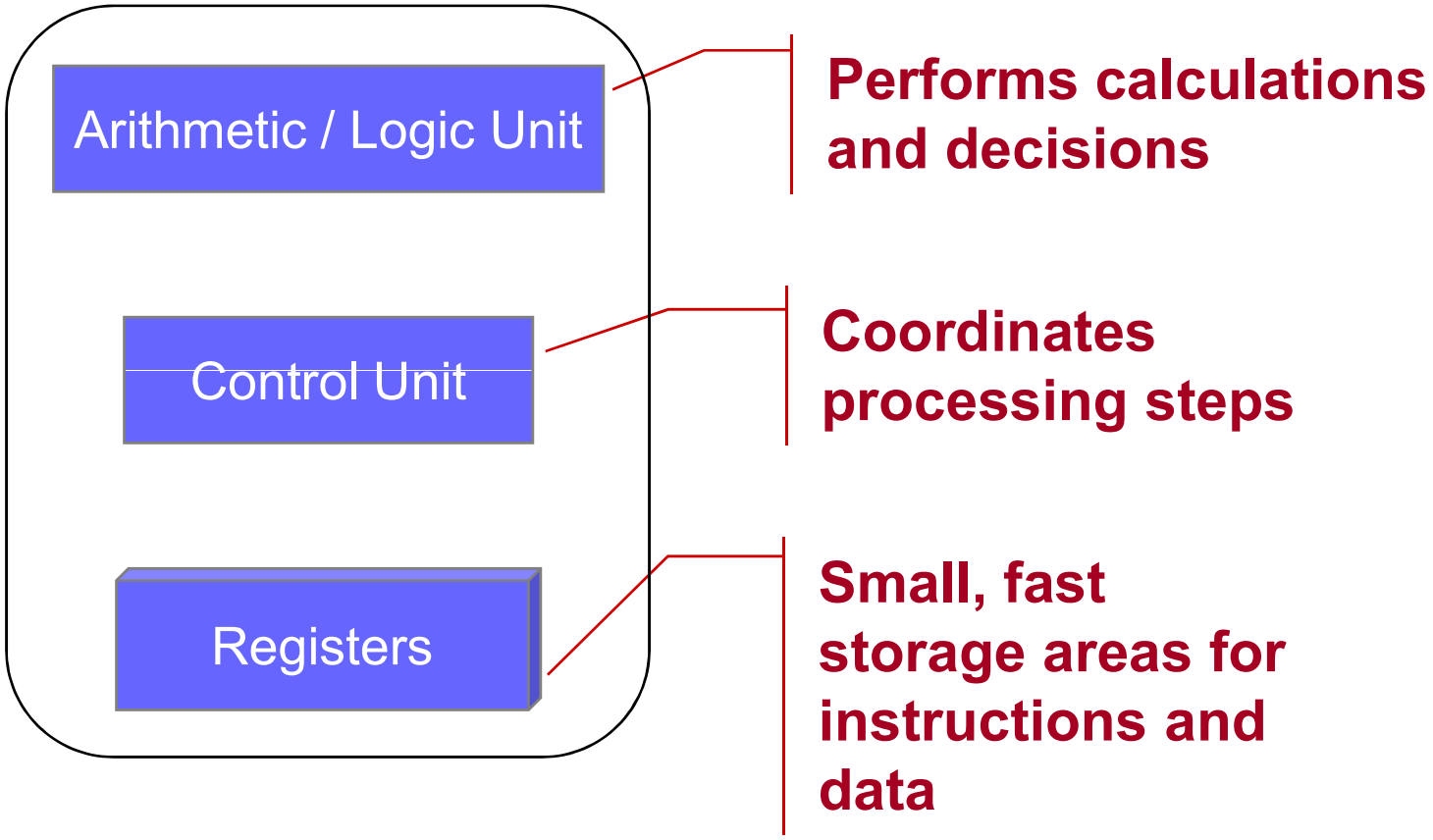
Inside The Microprocessor

- Internally, the microprocessor is made up of 3 main units.
  - The Arithmetic/Logic Unit (ALU)
  - The Control Unit.
  - An array of registers for holding data while it is being manipulated.

Organization of the Microprocessor

- The microprocessor can be divided into three main pieces:
  - **Arithmetic/Logic Unit** : Performs all computing and logic operations such as addition and subtraction as well as AND, OR and XOR.
  - **Register Array** : A collection of registers within the microprocessor itself. These are used primarily for data storage during program execution.
  - The number and the size of these registers differ from one microprocessor to the other.
  - **Control Unit** : As the name implies, the control Unit controls what is happening in the microprocessor.
  - It provides the necessary control and timing signals to all operations in the microprocessor as well as its contact to the outside world.

Central Processing Unit (CPU)



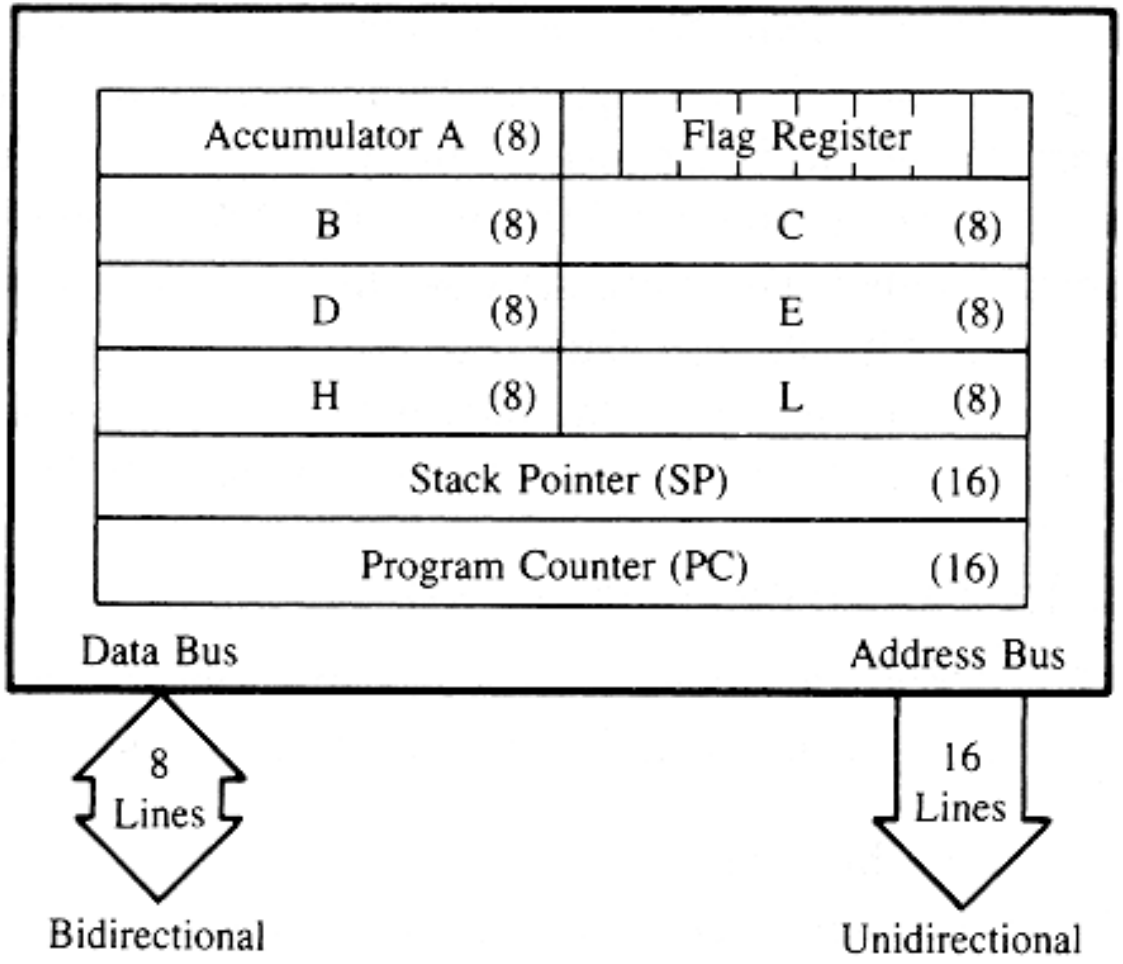
The 8085 Microprocessor

- The Intel 8085 is an 8-bit microprocessor introduced by Intel in 1977.
- It was binary compatible with the more-famous Intel 8080 but required less supporting hardware, thus allowing simpler and less expensive microcomputer systems to be built.
- The "5" in the model number came from the fact that the 8085 requires only a +5-Volt (V) power supply rather than the +5 V, −5 V and +12 V supplies the 8080 needed

The 8085 Microprocessor

- It is an 8-bit microprocessor with 16-bit address bus and hence can address up to 2^{16}= 65536 bytes (64KB) memory locations through A0–A15.
- The first 8 lines are of address bus and other 8 lines of data bus.
- A 16-bit program counter (PC), A 16-bit stack pointer (SP), Six 8-bit general purpose register arranged in pairs: BC, DE, HL.
- It requires a signal +5V power supply and operates at 3.2 MHZ single phase clock.
- It is enclosed with 40 pins DIP

The 8085: Registers



23

The 8085: CPU Internal Structure

Registers

- Six general purpose 8-bit registers: B, C, D, E, H, L
- They can also be combined as register pairs to perform 16-bit operations: BC, DE, HL
- Registers are programmable (data load, move, etc.)

Accumulator

- Single 8-bit register that is part of the ALU !
- Used for arithmetic / logic operations – the result is always stored in the accumulator.

The 8085: CPU Internal Structure

Flag Bits

- Indicate the result of condition tests.
- Carry, Zero, Sign, Parity, etc.
- Conditional operations (IF / THEN) are executed based on the condition of these flag bits.

- Program Counter (PC)
  - Contains the memory address (16 bits) of the instruction that will be executed in the next step.

Stack Pointer (SP)

- The stack pointer is also a 16-bit register that is used to point into memory. This register points to a special area called the stack.
- The stack is an area of memory used to hold data that will be retreived soon.The stack is usually accessed in a Last In First Out (LIFO) fashion.

Memory

- Memory stores information such as instructions and data in binary format (0 and 1). It provides this information to the microprocessor whenever it is needed.
- Usually, there is a memory “sub-system” in a microprocessor-based system. This sub-system includes:
  - The registers inside the microprocessor
  - Read Only Memory (ROM)
    - used to store information that does not change.
  - Random Access Memory (RAM) (also known as Read/Write Memory).
    - used to store information supplied by the user. Such as programs and data.

Memory

- To execute a program:
  - the user enters its instructions in binary format into the memory.
  - The microprocessor then reads these instructions and whatever data is needed from memory, executes the instructions and places the results either in memory or produces it on an output device.

I/O Input/Output

- Input and output devices are the system's means of communicating with the outside world. These devices are collectively known as *peripherals*.
  - **Input devices** transfer binary information from the outside world to the microprocessor.
    - Examples of input devices are: keyboard, mouse, bar code reader, scanner and the like.
  - **Output devices** transfer binary information from the microprocessor to the outside world.
    - Theses include things like an LED, a monitor, a printer and the like.

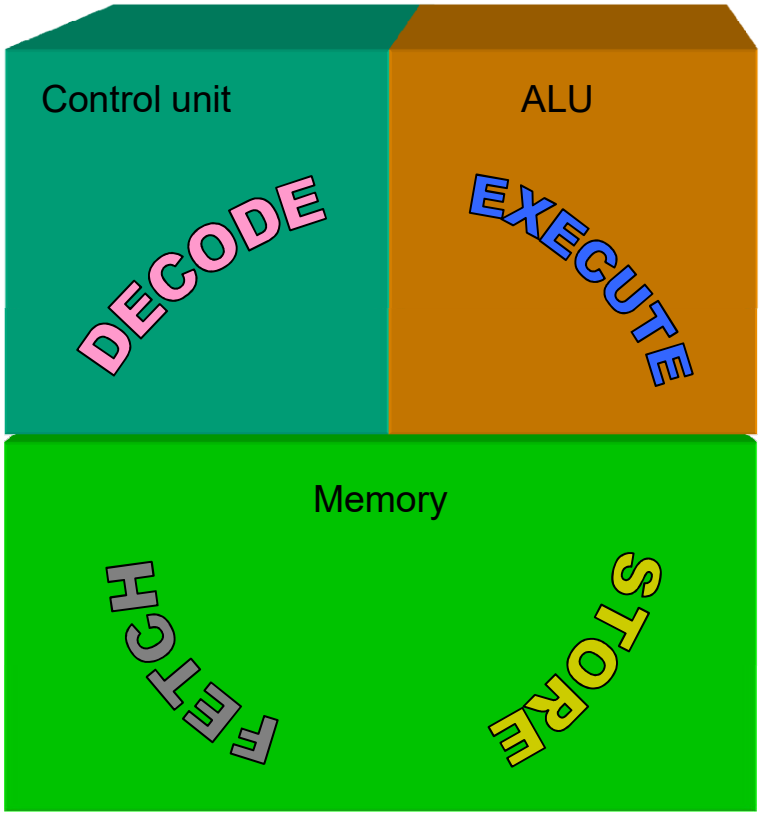
System Bus

- A communication path between the microprocessor and peripherals.
  - It is simply a group of wires carrying the voltages and curents representing the different bit values.
- The microprocessor communicates with only one peripheral at a time.
- Controlling the bus is done by the Control Unit.

The three cycle instruction execution model

- To execute a program, the microprocessor “reads” each instruction from memory, “interprets” it, then “executes” it.
- To use the right names for the cycles:
  - The microprocessor **fetches** each instruction,
  - **decodes** it,
  - Then **executes** it.
- This sequence is continued until all instructions are performed.

How a CPU works



The Control Unit

- The **control unit** manages four basic operations (fetch, decode, execute, and write-back)
  - The four-step process is known as the **machine cycle** or **processing cycle**
  - The processing cycle consists of two phases:
    - **Instruction Cycle**
      - **Fetch** – Gets the next program instruction from the computer's memory
      - **Decode** – Figures out what the program is telling the computer to do
    - **Execution Cycle**
      - **Execute** – Performs the requested action
      - **Write-back** (Store) – Writes (stores) the results to a register or to memory

Machine Language

- The number of bits that form the “word” of a microprocessor is fixed for that particular processor.
  - These bits define a maximum number of combinations.
    - For example an 8-bit microprocessor can have at most 2^8 = 256 different combinations.
- However, in most microprocessors, not all of these combinations are used.
  - Certain patterns are chosen and assigned specific meanings.
  - Each of these patterns forms an instruction for the microprocessor.
  - The complete set of patterns makes up the microprocessor’s machine language.

The 8085 Machine Language

- The 8085 (from Intel) is an 8-bit microprocessor.
  - The 8085 uses a total of 246 bit patterns to form its instruction set.
  - These 246 patterns represent only 74 instructions.
    - The reason for the difference is that some (actually most) instructions have multiple different formats.
  - Because it is very difficult to enter the bit patterns correctly, they are usually entered in hexadecimal instead of binary.
    - For example, the combination 0011 1100 which translates into “increment the number in the register called the accumulator”, is usually entered as 3C.

Assembly Language

- Entering the instructions using hexadecimal is quite easier than entering the binary combinations.
  - However, it still is difficult to understand what a program written in hexadecimal does.
  - So, each company defines a symbolic code for the instructions.
  - These codes are called “mnemonics”.
  - The mnemonic for each instruction is usually a group of letters that suggest the operation performed.

Assembly Language

- Using the same example from before,
  - 00111100 translates to 3C in hexadecimal
  - Its mnemonic is: “INR A”.
  - INR stands for “increment register” and A is short for accumulator.
- Another example is: 1000 0000,
  - Which translates to 80 in hexadecimal.
  - Its mnemonic is “ADD B”.
  - “Add register B to the accumulator and keep the result in the accumulator”.

Assembly Language

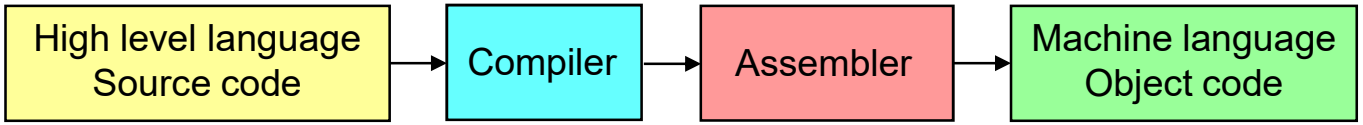
- It is important to remember that a machine language and its associated assembly language are completely machine dependent.
  - In other words, they are not transferable from one microprocessor to a different one.
- For example, Motorolla has an 8-bit microprocessor called the 6800.
  - The 8085 machine language is very different from that of the 6800. So is the assembly language.
  - A program written for the 8085 cannot be executed on the 6800 and vice versa.

“Assembling” The Program

- How does assembly language get translated into machine language?
  - There are two ways:
  - 1st there is “**hand assembly**”.
    - The programmer translates each assembly language instruction into its equivalent hexadecimal code (machine language). Then the hexadecimal code is entered into memory.
  - The other possibility is a program called an “**assembler**”, which does the translation automatically.

High Level Languages

- We said earlier that assembly and machine language are completely dependent on the microprocessor. They can not be easily moved from one to the other.
- To allow programs to be developed for multiple machines high level languages were developed.
  - These languages describe the operation of the program in general terms.
  - These programs are translated into microprocessor specific assembly language using a *compiler* or *interpreter* program.
    - These programs take as an input high level statements such as " I = j + k; " and translate them to machine language compatible with the microprocessor being used.

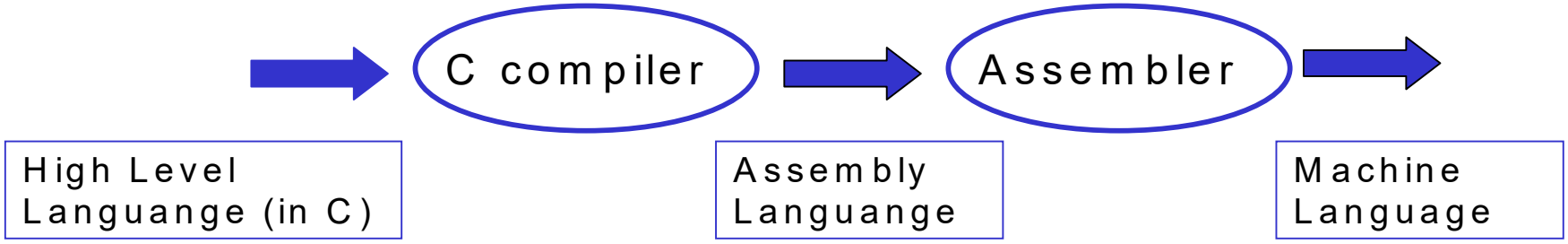


Differences

- High level language
- Assembly language
- Machine language

```
swap (int v[], int k)
{int temp;
        temp = v[k];
        v[k] = v[k+1];
        v[k+1] = temp;
}
```

```
swap;
        muli $2, $5,4        0001110001110 0
        add $2, $4,$2        0111100001110 0
        lw   $15, 0($2)      1111000011100 1
        lw   $16, 4($2)      0110101100100 1
```



Compiler vs. Interpreter

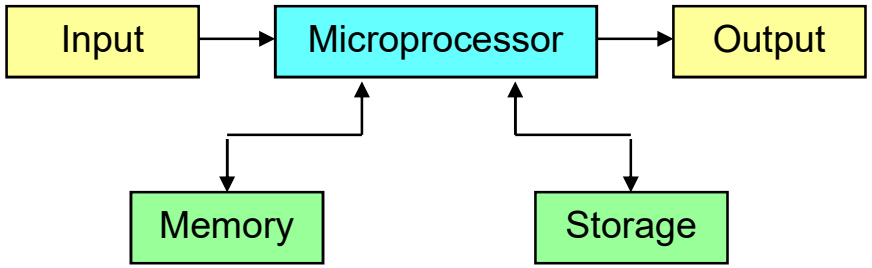
- What is the difference between Compiler and Interpreter?
  - A compiler translates the entire program at once and produces the object code.
  - An interpreter “ compiles “ the source code one line at-a-time.The object code for each line is produced, executed, and forgotten.
  - Each time the program is to be executed, it has to be re-interpreted.
    - Interpreters are very inefficient. Compilers produce object code that is quite a bit smaller and faster to execute.

Compiler vs. Interpreter

- Compilers are still inefficient when complete control is needed and when memory is very critical.
  - In such a situation, it is better to do the job yourself.
    - Writing a program in assembly language may not be easy.
    - But, you can control exactly how things are being done and which instruction is being used to perform each operation.

The Hardware/Software Interaction

- The hardware of a computer system is the collection of chips that make up its different pieces, including the microprocessor.
  - The hardware consists of five main systems:
    - The microprocessor
    - Memory (RAM & ROM)
    - Storage (Disk, CD)
    - Input Devices (keyboard, mouse)
    - Output Devices (monitor, printer).



The Hardware/Software interaction

- Software refers to any program that executes on the hardware.
  - It contains very low level programs that control the behavior of the hardware all the way to complicated applications like 3D graphics, video editing, and circuit simulation and design.

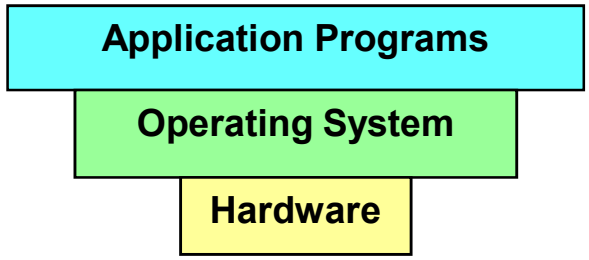
- The interaction between the two systems (hardware and software) is managed by a group of programs known collectively as *Operating system*.

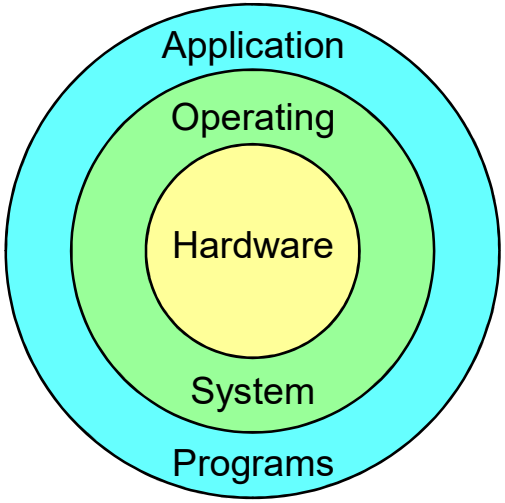
The Operating System

- The operating system is a layer between the application programs and the hardware.
  - Pieces of the operating system control the operation of the disks, the monitor, the keyboard, the mouse, the printer, the sound card, and even memory.
    - When a program wants to use one of these items, it sends a request to the operating system and the operating system in turn will perform the operation.
- When the computer is first turned on, the operating system starts to execute. It stays running as long as the computer is operating.

The Operating System

- The interaction of the user with the computer is through the operating system.
  - When the user invokes a program, the operating system starts the process and makes the program start executing. The program is executed on top of the operating system.





Operating Systems

- Examples of operating systems are:
  - MS-DOS
  - MS Windows
  - Macintosh OS
  - OS/2
  - UNIX
- Most operating systems are hardware specific:
  - For example, windows only runs on microprocessors made by Intel or those that behave the same way (i.e. “compatible”).
- Other operating systems (like UNIX) are designed to work on any platform (hardware).

Microprocessor Architecture

Microprocessor Architecture

- The microprocessor can be programmed to perform functions on given data by writing specific instructions into its memory.
  - The microprocessor reads one instruction at a time, matches it with its instruction set, and performs the data manipulation specified.
  - The result is either stored back into memory or displayed on an output device.
- **Operation Types in a Microprocessor**
  - All of the operations of the microprocessor can be classified into one of three types:
    - Microprocessor Initiated Operations
    - Internal Operations
    - Peripheral Initiated Operations

Microprocessor - Initiated Operations

- Microprocessor Performs primarily four operations as a part of communication process between MPU and peripheral devices.

1. Memory Read : Reads data (instructions) from memory
2. Memory Write : Writes data (instructions) in to memory
3. I/O Read : Accepts data from input devices.
4. I/O Write : Sends data to output devices

- To communicate with a peripheral, the MPU need to perform following steps :-

1. Identify the peripherals or the memory location (with its address)
2. Transfer binary information (data and instructions)
3. Provide timing or synchronization signals (control signals)

Microprocessor Initiated Operations

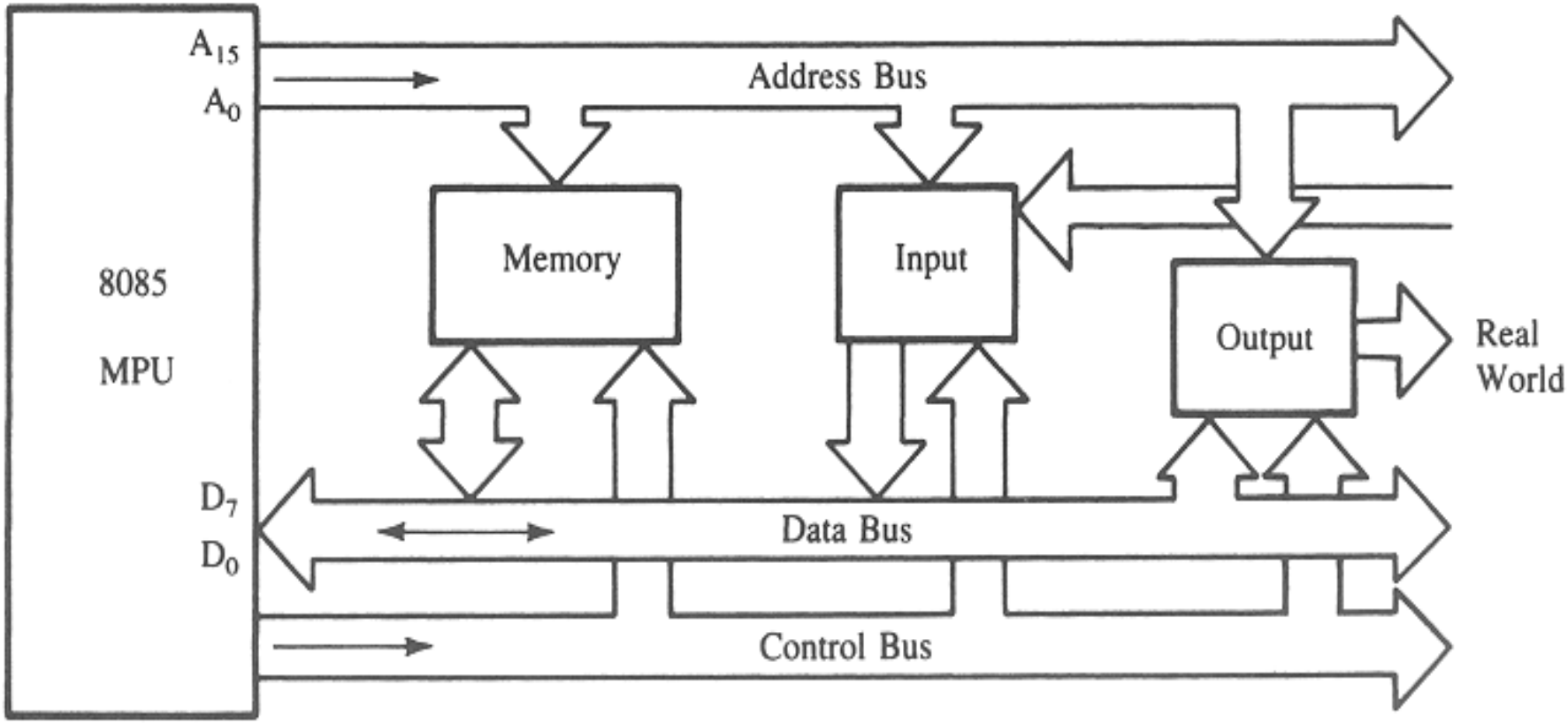
- It is important to note that the microprocessor treats memory and I/O devices the same way.
  - Input and output devices simply look like memory locations to the microprocessor.
    - For example, the keyboard may look like memory address A3F2H. To get what key is being pressed, the microprocessor simply reads the data at location A3F2H.

The 8085 Architecture

- The 8085 uses three separate busses to perform its operations
  - The **Address bus**.
  - The **Data bus.**
  - The **Control bus**.

The 8085 Bus Structure

- The 8-bit 8085 CPU (or MPU – Micro Processing Unit) communicates with the other units using a 16-bit address bus, an 8-bit data bus and a control bus.



The Address Bus

- 16 bits wide (A_0 A_1…A_{15})
  - Therefore, the 8085 can access locations with numbers from 0 to 65,536. Or, the 8085 can access a total of 64K addresses.
- “Unidirectional”.
  - Information flows out of the microprocessor and into the memory or peripherals.
- When the 8085 wants to access a peripheral or a memory location, it places the 16-bit address on the address bus and then sends the appropriate control signals.

The Data Bus and Control Bus

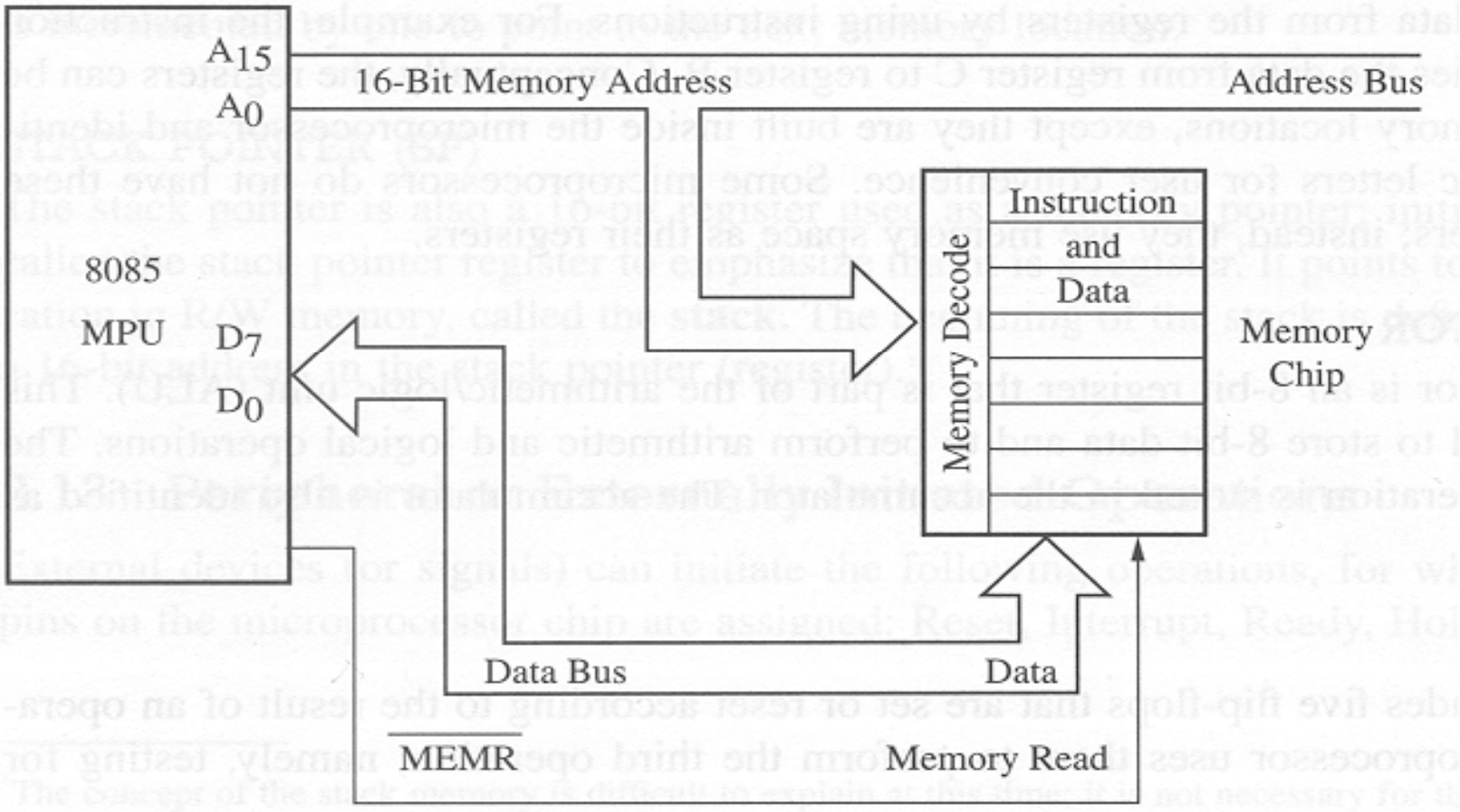
- Data bus :
  - 8 bits wide (D_0 D_1…D_7)
  - “Bi-directional”.
    - Information flows both ways between the microprocessor and memory or I/O.
  - The 8085 uses the data bus to transfer the binary information.
  - Since the data bus has 8-bits only, then the 8085 can manipulate data 8 bits at-a-time only.
- Control bus :
  - There is no real control bus. Instead, the control bus is made up of a number of single bit control signals.

The 8085: CPU Internal Structure

The internal architecture of the 8085 CPU is capable of performing the following operations:

- Store 8-bit data (Registers, Accumulator)
- Perform arithmetic and logic operations (ALU)
- Test for conditions (IF / THEN)
- Sequence the execution of instructions
- Store temporary data in RAM during execution

Memory Read Operation

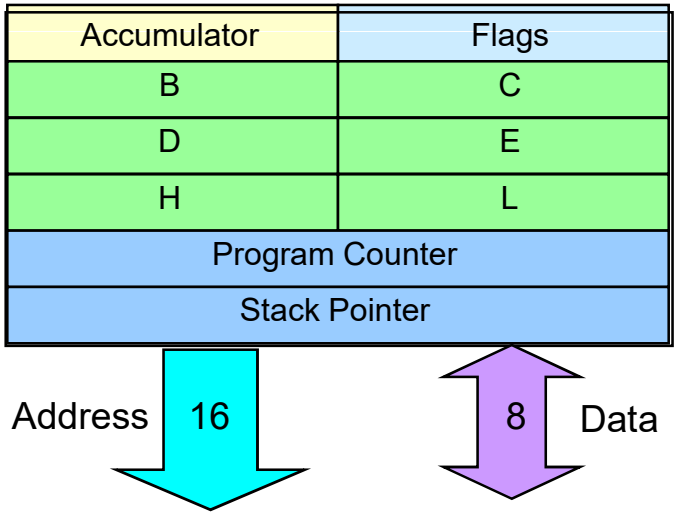


The Read Operation

- To read the contents of a memory location, the following steps take place:
  - The microprocessor places the 16-bit address of the memory location on the address bus.
  - The microprocessor activates a control signal called “memory read” which enables the memory chip.
  - The memory decodes the address and identifies the right location.
  - The memory places the contents on the data bus.
  - The microprocessor reads the value of the data bus after a certain amount of time.

Internal Data Operations

- The 8085 can perform a number of internal operations. Such as: storing data, Arithmetic & Logic operations, Testing for condition, etc.
  - To perform these operations, the microprocessor needs an internal architecture similar to the following:



The Internal Architecture

- Registers
  - Six general purpose 8-bit registers: B, C, D, E, H, L
  - They can also be combined as register pairs to perform 16-bit operations: BC, DE, HL
  - Registers are programmable (data load, move, etc.)
- Accumulator
  - Single 8-bit register that is part of the ALU
  - Used for arithmetic / logic operations – the result is always stored in the accumulator.
- Flag Bits
  - Indicate the result of condition tests.
  - Carry, Zero, Sign, Parity, Auxiliary Carry etc.
  - Conditional operations (IF / THEN) are executed based on the condition of these flag bits.

The Flags register

- There is also the flags register whose bits are affected by the arithmetic & logic operations.
  - S-sign flag : The sign flag is set if bit D7 of the accumulator (result) = 1; otherwise it is reset.
  - Z-zero flag : Set to 1 when the result of the ALU operation is 0. Otherwise is reset. This flag is affected by operations on the accumulator as well as other registers.
  - AC-Auxiliary Carry : This flag is set when a carry is generated from bit D3 and passed to D4 . This flag is used only internally for BCD operations.
  - P-Parity flag : After an ALU operation if the result has an even no. of 1's the p-flag is set. Otherwise it is reset.
  - CY-carry flag : If an arithmetic operation results in a carry, the CY flag is set; otherwise it is reset.

The Internal Architecture

- The Program Counter (PC)
  - This is a register that is used to control the sequencing of the execution of instructions.
  - This register always holds the address of the next instruction.
  - Since it holds an address, it must be 16 bits wide.
- The Stack pointer
  - The stack pointer is also a 16-bit register that is used to point into memory.
  - The memory this register points to is a special area called the stack.
  - The stack is an area of memory used to hold data that will be retreived soon.
  - The stack is usually accessed in a Last In First Out (LIFO) fashion.

Externally Initiated Operations

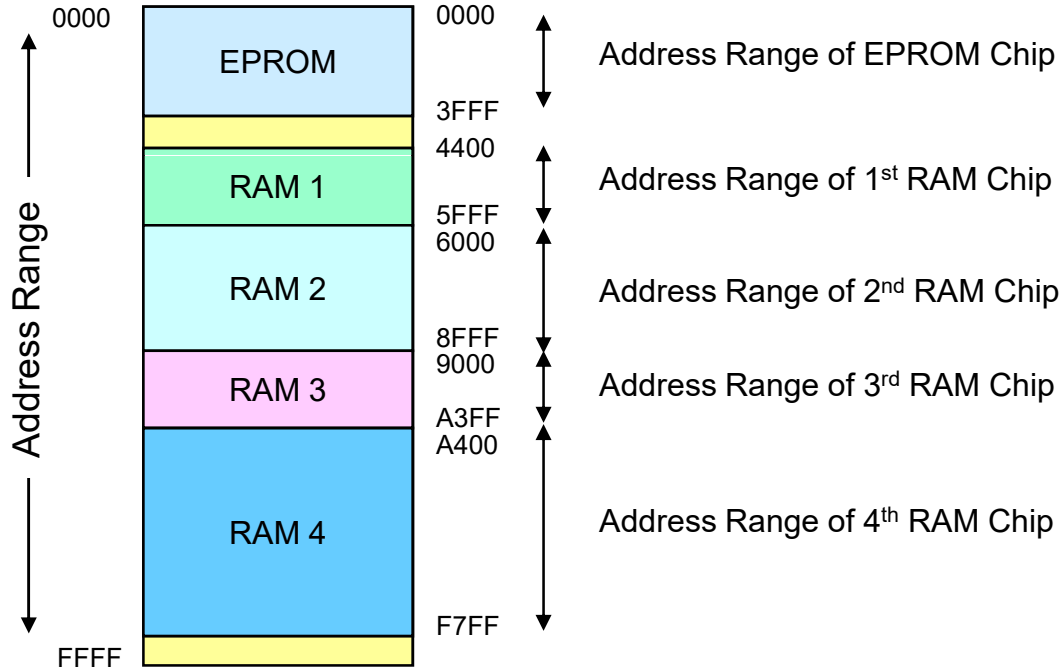
- External devices can initiate (start) one of the 4 following operations:
  - **Reset**
    - All operations are stopped and the program counter is reset to 0000.
  - **Interrupt**
    - The microprocessor's operations are interrupted and the microprocessor executes what is called a "**service routine**".
    - This routine "handles" the interrupt, (perform the necessary operations).
    - Then the microprocessor returns to its previous operations and continues.

Externally Initiated Operations

- **Ready**
  - The 8085 has a pin called RDY. This pin is used by external devices to stop the 8085 until they catch up.
  - As long as the RDY pin is low, the 8085 will be in a wait state.
- **Hold**
  - The 8085 has a pin called HOLD. This pin is used by external devices to gain control of the busses.
  - When the HOLD signal is activated by an external device, the 8085 stops executing instructions and stops using the buses.
  - This would allow external devices to control the information on the buses. Example **DMA**.

Memory Map and Addresses

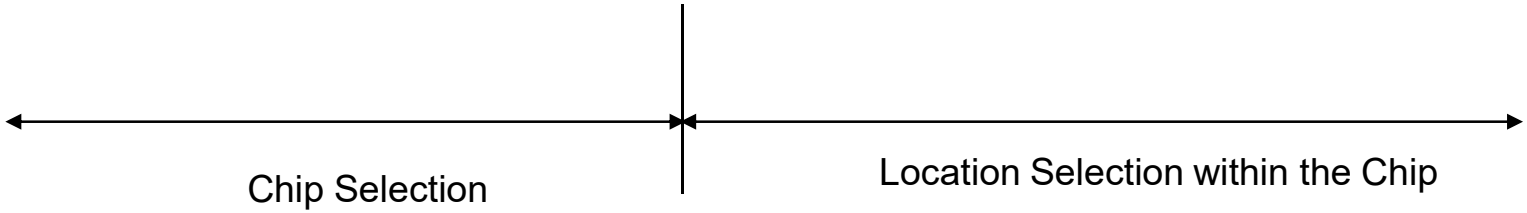
- The memory map is a picture representation of the address range and shows where the different memory chips are located within the address range.



The 8085 and Address Ranges

- Now, we can break up the 16-bit address of the 8085 into two pieces:

$A_{15}\ A_{14}\ A_{13}\ A_{12}\ A_{11}\ A_{10}\ A_9\ A_8\ A_7\ A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$



  - Depending on the combination on the address lines A_{15} - A_{10} , the address range of the specified chip is determined.

High-Order vs. Low-Order Address Lines

- The address lines from a microprocessor can be classified into two types:
  - High-Order
    - Used for memory chip selection
  - Low-Order
    - Used for location selection within a memory chip.
  - This classification is highly dependent on the memory system design.

Addressing Modes in 8085

- These are the instructions used to transfer the data from one register to another register, from the memory to the register, and from the register to the memory without any alteration in the content.
- Addressing modes in 8085 is classified into 5 groups:
  - Immediate addressing mode
  - Register addressing mode
  - Direct addressing mode
  - Indirect addressing mode
  - Implied addressing mode

Addressing Modes in 8085

- **Immediate addressing mode**
- In this mode, the 8/16-bit data is specified in the instruction itself as one of its operand. **For example:** MVI K, 20F: means 20F is copied into register K.
- **Register addressing mode**
- In this mode, the data is copied from one register to another. **For example:** MOV K, B: means data in register B is copied to register K.

Addressing Modes in 8085

- **Direct addressing mode**
- In this mode, the data is directly copied from the given address to the register. **For example:** LDB 5000K: means the data at address 5000K is copied to register B.
- **Indirect addressing mode**
- In this mode, the data is transferred from one register to another by using the address pointed by the register. **For example:** MOV K, B: means data is transferred from the memory address pointed by the register to the register K.

Addressing Modes in 8085

- **Implied addressing mode**
- This mode doesn't require any operand; the data is specified by the opcode itself. **For example:** CMA (finds and stores the 1's complement of the contains of accumulator A in A).

8085 instruction set

- **8085 instruction set consists of the following instructions:**
  - Data moving instructions.
  - Arithmetic - add, subtract, increment and decrement.
  - Logic - AND, OR, XOR and rotate.
  - Control transfer - conditional, unconditional, call subroutine, return from subroutine and restarts.
  - Input/Output instructions.
  - Other - setting/clearing flag bits, enabling/disabling interrupts, stack operations, etc

The 8085 Microprocessor Architecture

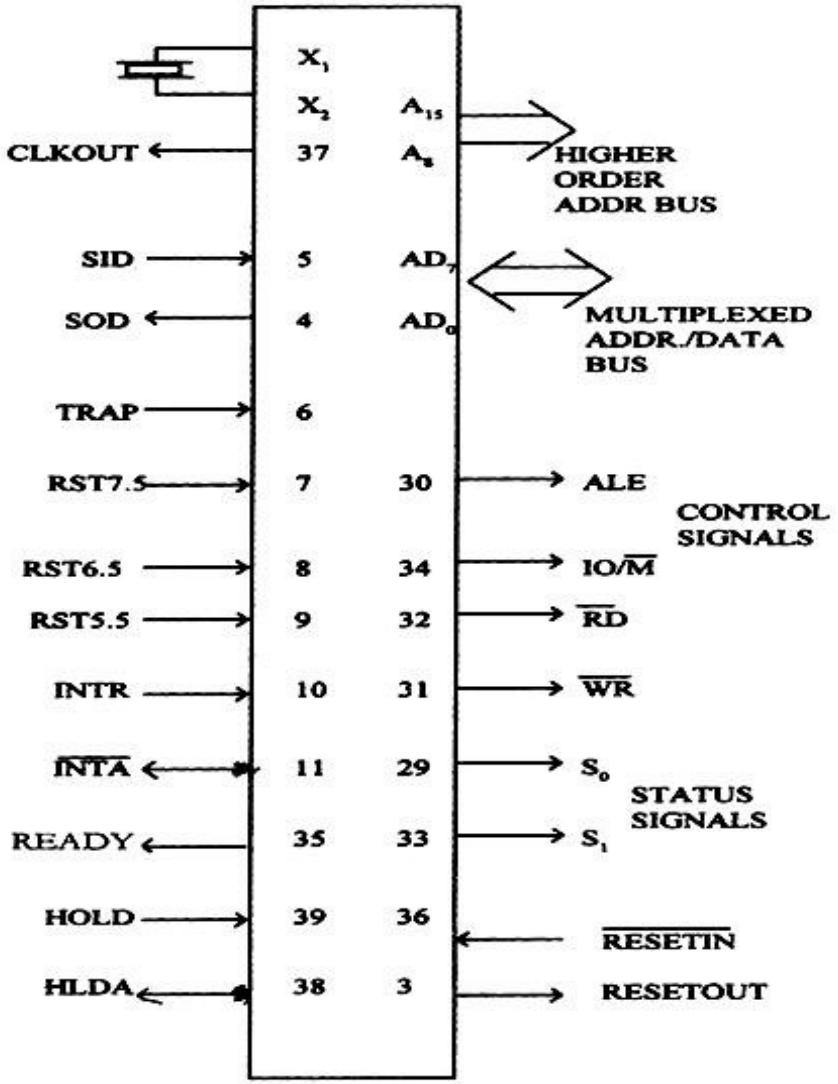
Introduction

- It provides 16 address lines so it can access 2^16 =64K bytes of memory.
- It generates 8 bit I/O address so it can access 2^8=256 input ports.
- It provides 5 hardware interrupts: TRAP, RST 5.5, RST 6.5, RST 7.5,INTR.
- It provides Accu ,one flag register, 6 general purpose registers and two special purpose registers(SP,PC).
- It provides serial lines SID ,SOD. So serial peripherals can be interfaced with 8085 directly.

The 8085 and Its Busses

- The 8085 is an 8-bit general purpose microprocessor that can address 64K Byte of memory.
- It has 40 pins and uses +5V for power. It can run at a maximum frequency of 3 MHz.
  - The pins on the chip can be grouped into 6 groups:
    - Address Bus.
    - Data Bus.
    - Control and Status Signals.
    - Power supply and frequency.
    - Externally Initiated Signals.
    - Serial I/O ports.

8085 Pinout Control and Status signals



8085 pin description

Some important pins are :

- AD0-AD7: Multiplexed Address and data lines.
- A8-A15: Tri-stated higher order address lines.
- ALE: Address latch enable is an output signal.It goes high when operation is started by processor .
- S0,S1: These are the status signals used to indicate type of operation.
- $\overline{RD}$: Read is active low input signal used to read data from I/O device or memory.
- $\overline{WR}$:Write is an active low output signal used to write data on memory or an I/O device.

8085 pin description

- READY: This an output signal used to check the status of output device. If it is low, µP will WAIT until it is high.
- TRAP: It is an Edge triggered highest priority , non mask able interrupt. After TRAP, restart occurs and execution starts from address 0024H.
- RST5.5,6.5,7.5:These are maskable interrupts and have low priority than TRAP.
- $\overline{\text{INTR}}$&INTA:INTR is a interrupt request signal after which µP generates INTA or interrupt acknowledge signal.
- IO/$\overline{\text{M}}$:This is output pin or signal used to indicate whether 8085 is working in I/O mode(IO/$\overline{\text{M}}$=1) or Memory mode(IO/$\overline{\text{M}}$=0).

8085 pin description

- HOLD&HLDA:HOLD is an input signal .When µP receives HOLD signal it completes current machine cycle and stops executing next instruction. In response to HOLD µP generates HLDA that is HOLD Acknowledge signal.
- RESET $\overline{\text{IN}}$:This is input signal. When RESET $\overline{\text{IN}}$ is low µp restarts and starts executing from location 0000H.
- SID: Serial input data is input pin used to accept serial 1 bit data .
- X1X2 :These are clock input signals and are connected to external LC,or RC circuit. These are divide by two so if 6 MHz is connected to X1X2, the operating frequency becomes 3 MHz.
- VCC&VSS:Power supply VCC=+ -5Volt& VSS=-GND reference.

The Address and Data Busses

- The address bus has 8 signal lines A8 – A15 which are unidirectional.
- The other 8 address bits are multiplexed (time shared) with the 8 data bits.
  - So, the bits AD0 – AD7 are bi-directional and serve as A0 – A7 and D0 – D7 at the same time.
    - During the execution of the instruction, these lines carry the address bits during the early part, then during the late parts of the execution, they carry the 8 data bits.
  - In order to separate the address from the data, we can use a latch to save the value before the function of the bits changes.

The Control and Status Signals

- There are 4 main control and status signals. These are:
  - ALE: Address Latch Enable. This signal is a pulse that become 1 when the AD0 – AD7 lines have an address on them. It becomes 0 after that. This signal can be used to enable a latch to save the address bits from the AD lines.
  - RD: Read. Active low.
  - WR: Write. Active low.
  - IO/M: This signal specifies whether the operation is a memory operation (IO/M=0) or an I/O operation (IO/M=1).
  - S1 and S0 : Status signals to specify the kind of operation being performed .Usually un-used in small systems.

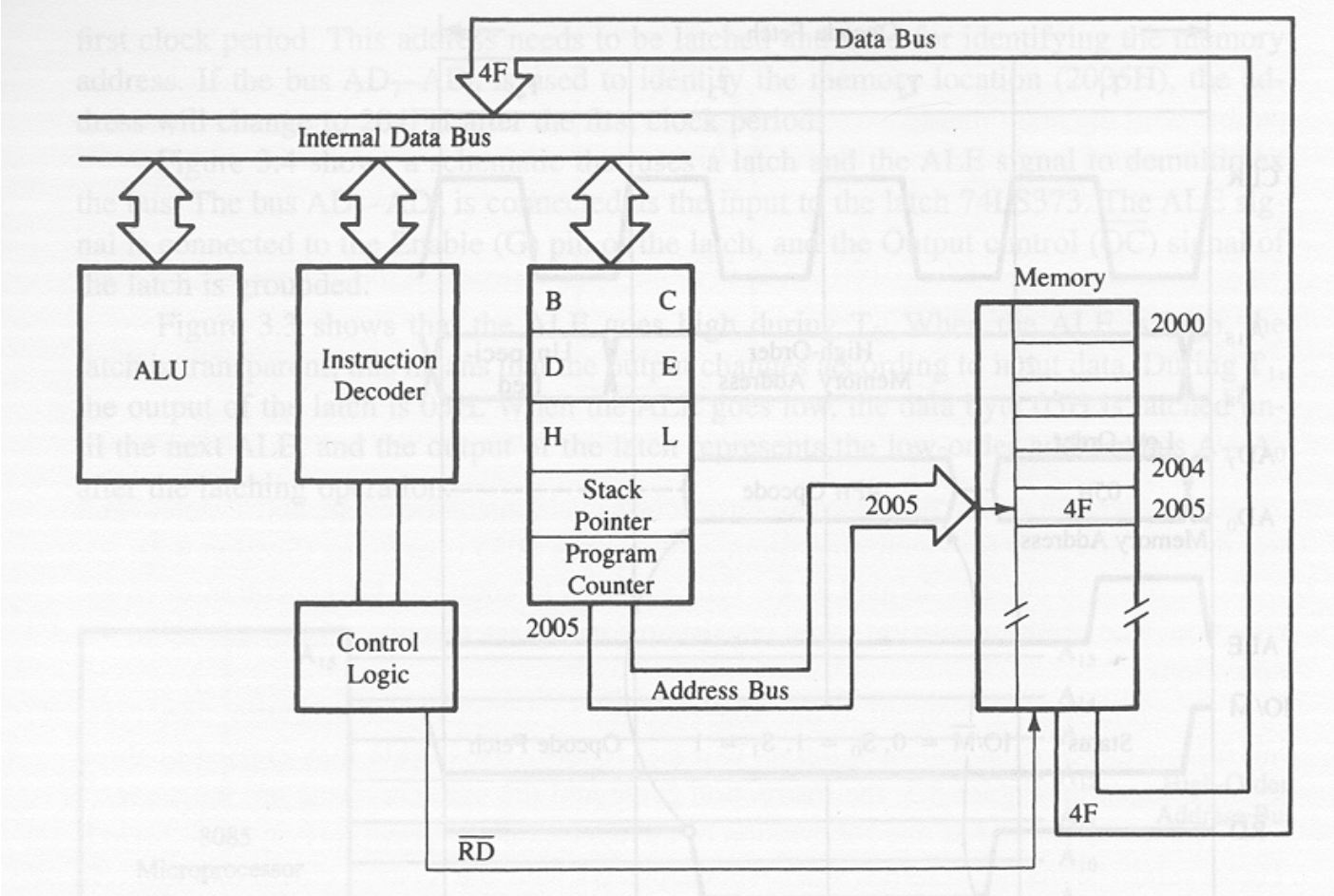
Microprocessor Communication and Bus Timing

- To understand how the microprocessor operates and uses these different signals, we should study the process of communication between the microprocessor and memory during a memory read or write operation.
- Lets look at timing and the data flow of an instruction fetch operation.

Steps For Fetching an Instruction

- Lets assume that we are trying to fetch the instruction at memory location 2005. That means that the program counter is now set to that value.
  - The following is the sequence of operations:
    - The program counter places the address value on the address bus and the controller issues a RD signal.
    - The memory's address decoder gets the value and determines which memory location is being accessed.
    - The value in the memory location is placed on the data bus.
    - The value on the data bus is read into the instruction decoder inside the microprocessor.
    - After decoding the instruction, the control unit issues the proper control signals to perform the operation.

Example: Instruction Fetch Operation



Cycles and States

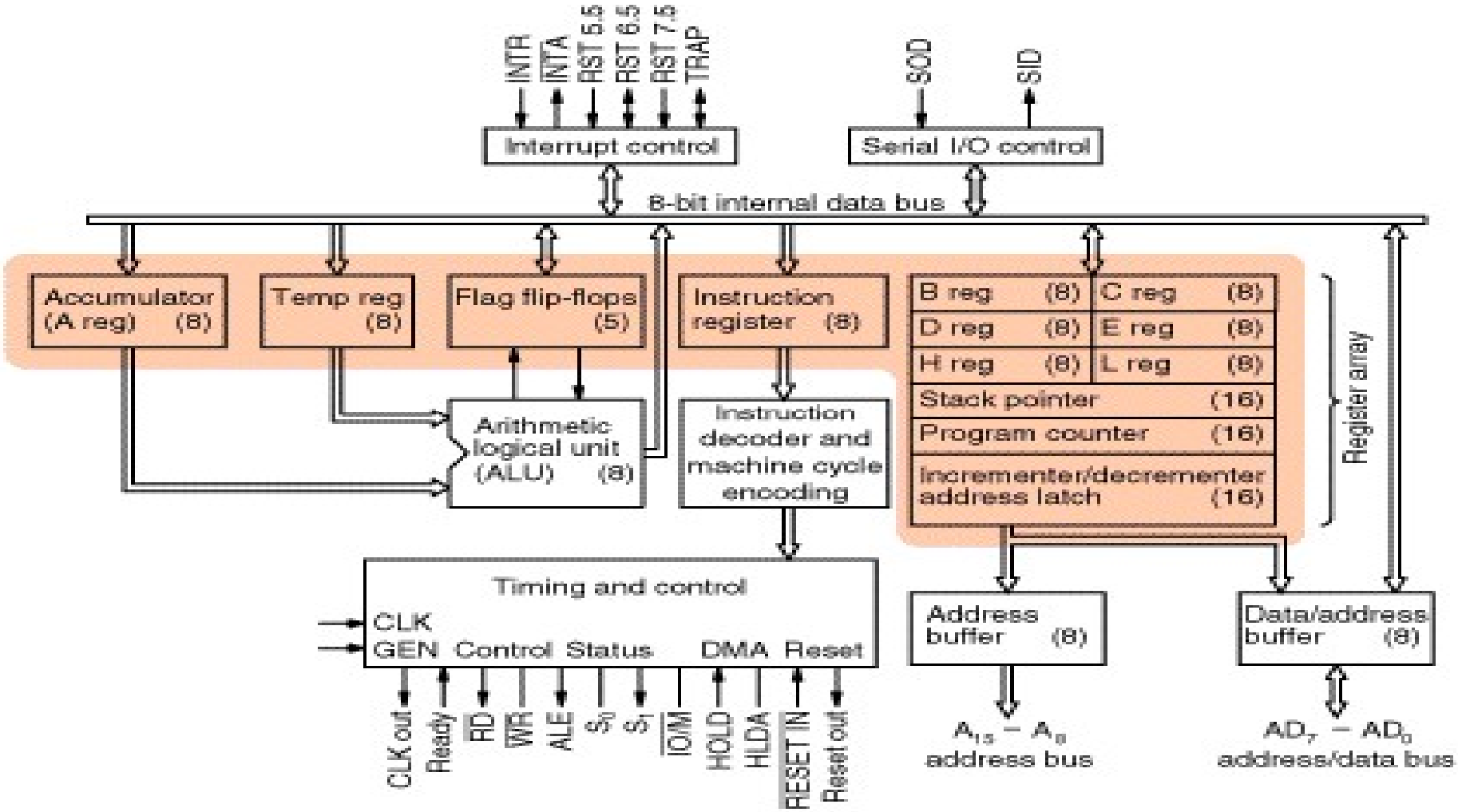
- From the above discussion, we can define terms that will become handy later on:
  - T- State: One subdivision of an operation. A T-state lasts for one clock period.
    - An instruction's execution length is usually measured in a number of T-states. (clock cycles).
  - Machine Cycle: The time required to complete one operation of accessing memory, I/O, or acknowledging an external request.
    - This cycle may consist of 3 to 6 T-states.
  - Instruction Cycle: The time required to complete the execution of an instruction.
    - In the 8085, an instruction cycle may consist of 1 to 6 machine cycles.

A closer look at the 8085 Architecture

- Previously we discussed the 8085 from a programmer's perspective.

- Now, lets look at some of its features with more detail.

A closer look at the 8085 Architecture



The ALU

- In addition to the arithmetic & logic circuits, the ALU includes the accumulator, which is part of every arithmetic & logic operation.
- Also, the ALU includes a temporary register used for holding data temporarily during the execution of the operation. This temporary register is not accessible by the programmer.

More on the 8085 machine cycles

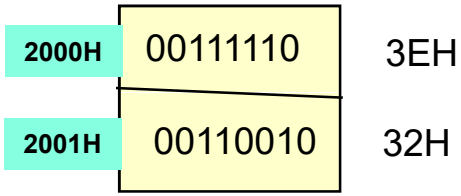
- The 8085 executes several types of instructions with each requiring a different number of operations of different types. However, the operations can be grouped into a small set.
- The three main types are:
  - Memory Read and Write.
  - I/O Read and Write.
  - Request Acknowledge.
- These can be further divided into various operations (machine cycles).

Opcode Fetch Machine Cycle

- The first step of executing any instruction is the Opcode fetch cycle.
  - In this cycle, the microprocessor brings in the instruction's Opcode from memory.
    - To differentiate this machine cycle from the very similar "memory read" cycle, the control & status signals are set as follows:
      - IO/M=0, s0 and s1 are both 1.
  - This machine cycle has four T-states.
    - The 8085 uses the first 3 T-states to fetch the opcode.
    - T4 is used to decode and execute it.
  - It is also possible for an instruction to have 6 T-states in an opcode fetch machine cycle.

The Memory Read Machine Cycle

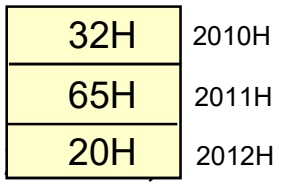
- The memory read machine cycle is exactly the same as the opcode fetch except:
  - It only has 3 T-states. The s0 signal is set to 0 instead.
  - To understand the memory read machine cycle, let's study the execution of the following instruction: **MVI A, 32H** (Load byte 32H in accumulator)



  - In memory, this instruction looks like:
    - The first byte 3EH represents the opcode for loading a byte into the accumulator (MVI A), the second byte is the data to be loaded.

Machine Cycles vs. Number of bytes in the instruction

- Machine cycles and instruction length, do not have a direct relationship.
  - To illustrate lets look at the machine cycles needed to execute the following instruction.
    - STA 2065H
    - This is a 3-byte instruction requiring 4 machine cycles and 13 T-states.
    - The machine code will be stored in memory as shown to the right



    - This instruction requires the following 4 machine cycles:
      - Opcode fetch to fetch the opcode (32H) from location , decode it and determine that 2 more bytes are needed (4 T-states).
      - Memory read to read the low order byte of the address (65H) (3 T-states).
      - Memory read to read the high order byte of the address (20H) (3 T-states).
      - A memory write to write the contents of the accumulator into the memory location.

Address decoding

- The result of address decoding is the identification of a register for a given address.
  - A large part of the address bus is usually connected directly to the address inputs of the memory chip.
  - This portion is decoded internally within the chip.
  - What concerns us is the other part that must be decoded externally to select the chip.
  - This can be done either using logic gates or a decoder.

Instruction Cycle

- The main function of CPU is to execute the program.
- A program consists of sequence of instruction to perform a particular task. A program is stored in memory.
- CPU fetches one instruction at a time from the memory and execute it.
- The CPU repeats this process till it executes all the instructions of the program. Then takes another program to execute.
- Instruction cycle : The necessary steps that the processor has to carry out for fetching an instruction from memory and executing it. It consist of two parts
  - A fetch cycle : Here the CPU fetches the machine code of the instruction (opcode) from the memory. The necessary steps that are carried out to fetch an opcode from the memory, constitute a fetch cycle.
  - A execute cycle: Here instruction are executed.the necessary steps which are carried to execute an instruction, constitute an execute cycle.

Registers and Busses

- Instruction Pointer (IP)
  - The location of the **next** instruction.
  - Sometimes called the Program Counter (PC)
- Memory Access Register (MAR)
- Current Instruction Register (CIR) or (IR)
  - The **current** instruction.
- Accumulator (AX)
  - Used for short term storage, and in many instructions
- Address Bus
  - Moves locations of data to different registers, particularly between the Instruction Pointer, Memory Address Register, and Memory
- Data Bus
  - Moves contents of memory addresses

Instruction Cycle (Fetch Operation)

- To fetch an opcode from a memory location the following steps are performed :
  - The program counter places the address of the memory location in which the opcode is stored, on the address bus.
  - The CPU sends the required memory control signals so as to enable the memory to send the opcode.
  - The opcode stored in the memory location is placed on the data bus and transferred to the CPU.
- The above steps required 3 clock cycles. If memory is slow the time taken may be more. In that case the CPU has to wait for some time till the memory transfer the opcode to the CPU.
- The extra clock cycle for which the CPU waits are known as *wait cycles*.
- Most of the microprocessors have circuitry to introduce wait cycles to cope with slow memory.

Instruction Cycle (Execute Operation)

- The opcode which is fetched from the memory is placed first of all in the data/address buffer/register.
- Thereafter if goes to the instruction register.
- From the instruction register it goes to the decoder circuitry which is within the CPU.
- The decoder circuitry decodes the opcode.
- After the opcode is decoded the CPU comes to know what operation is to be performed, and then execution begins.
- If the operand is in the general purpose regesiter, the execution is immediately performed. In such a situation the time required for decoding and executing the instruction is only 1 clock cycle.
- If the required data or operand address is still in the memory, the CPU reads them from the memory.

Instruction Cycle (Execute Operation)

- For reading data or operand address from the memory the CPU performs read operation. The read cycle is similar to an opcode fetch cylce. In a read cycle the quantity received from the memory is data or address instead of opcode.
- After receiving data from the memory, CPU performs execution operation.
- Some instruction may require write operation. In a write cycle data are transferred from the CPU to the memory or an outpur device.
- Thus an execute cycle may involve one or more read or write cycles or both.

Instruction Cycle (Machine Cycle and State)

- Machine cycle : The necessayr steps which are carried out to acces a memory or I/O device, constitute a machine cycle.. In other words necessary steps which are carried out to perform a fetch, read or write operation, constitute a machine cycle.
- An instruction cycle consists of a number of machine cycles.
- In one machine cycle only one operation such as opcode fetch, memory read, memory write, I/O read or I/O write is performed.

Instruction Cycle (Machine Cycle and State)

- The first machine cylce of instruction cycle is an opcode fetch cycle.
- The single-byte instruction are executed in only one machine cycle.
- Two-byte and 3 byte instruction need more machine cycles as additional machine cyles are required for reading /writing data from/into the memory or I/O devices.
- A state (or T- state) is one subdivision of an operation performed in one clock period. So one clock cycle of the system clock is reffered to as a state.