IIT KHARAGPUR

NPTEL ONLINE
CERTIFICATION COURSES

8085 Microprocessor

Santanu Chattopadhyay

Electronics and Electrical Communication Engineering

Basic Concepts of Microprocessors

- Differences between:
  - Microcomputer – a computer with a microprocessor as its CPU. Includes memory, I/O etc.
  - Microprocessor – silicon chip which includes ALU, register circuits & control circuits
  - Microcontroller – silicon chip which includes microprocessor, memory & I/O in a single package.

What is a Microprocessor?

- The word comes from the combination micro and processor.
  - Processor means a device that processes whatever. In this context processor means a device that processes numbers, specifically binary numbers, 0's and 1's.
    - To process means to manipulate. It is a general term that describes all manipulation. Again in this content, it means to perform certain operations on the numbers that depend on the microprocessor's design.

What about micro?

- 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.

Was there ever a “mini”- processor?

- No.
  - – It went directly from discrete elements to a single chip. However, comparing today’s microprocessors to the ones built in the early 1970’s you find an extreme increase in the amount of integration.

- So, What is a microprocessor?

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.

Definition (Contd.)

- 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. This group of operations is 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.

Definition (Contd.)

- Numbers: The microprocessor has a very narrow view on life. It only understands binary numbers.
- A binary digit is called a bit (which comes from **b**inary dig**it**).
- 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.
  - 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, 80, 128 bits

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.)

- Stored in memory :
  - First, what is memory?
    - Memory is the location where information is kept while not in current use.
    - Memory is usually measured by the number of bytes it can hold.
    - KB, MB, GB etc.

Definition (Contd.)

– Stored in memory:

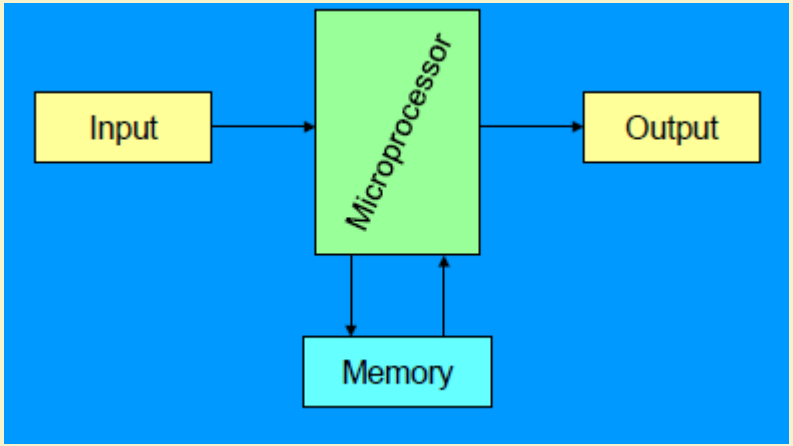
- When a program is entered into a computer, it is stored in memory. Then as the microprocessor starts to execute the instructions, it brings the instructions from memory one at a time.
- Memory is also used to hold the data.
  - The microprocessor reads (brings in) the data from memory when it needs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

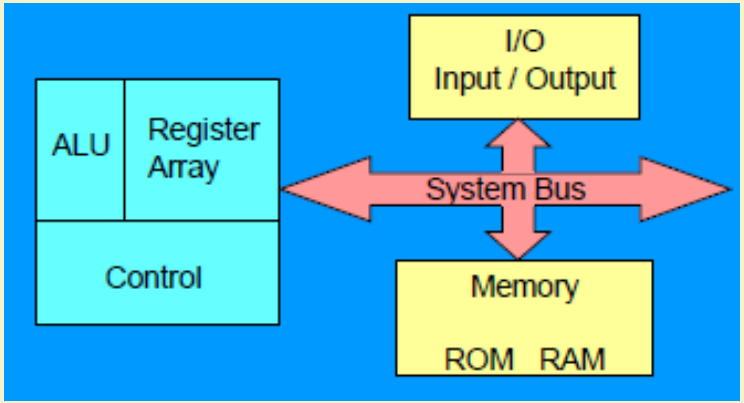


Inside The Microprocessor

- 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 a microprocessor- based system

- Let’s expand the picture a bit.

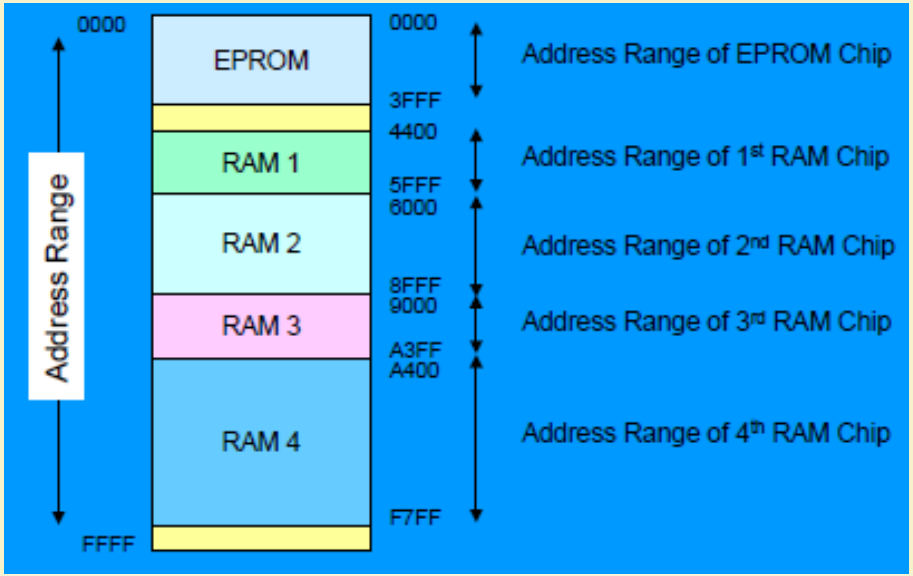


Memory

- 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: Registers, ROM, RAM

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.



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.

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.

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.
- 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.
  - 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 (OPCODE)
  - 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.
- 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.

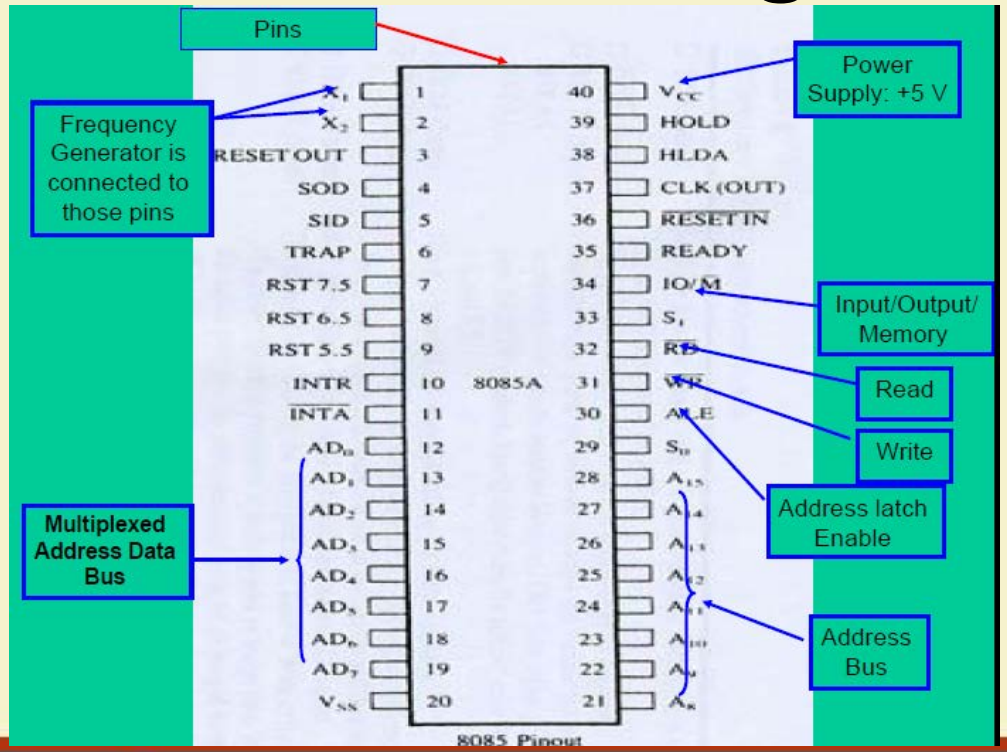
“Assembling” The Program

- 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.

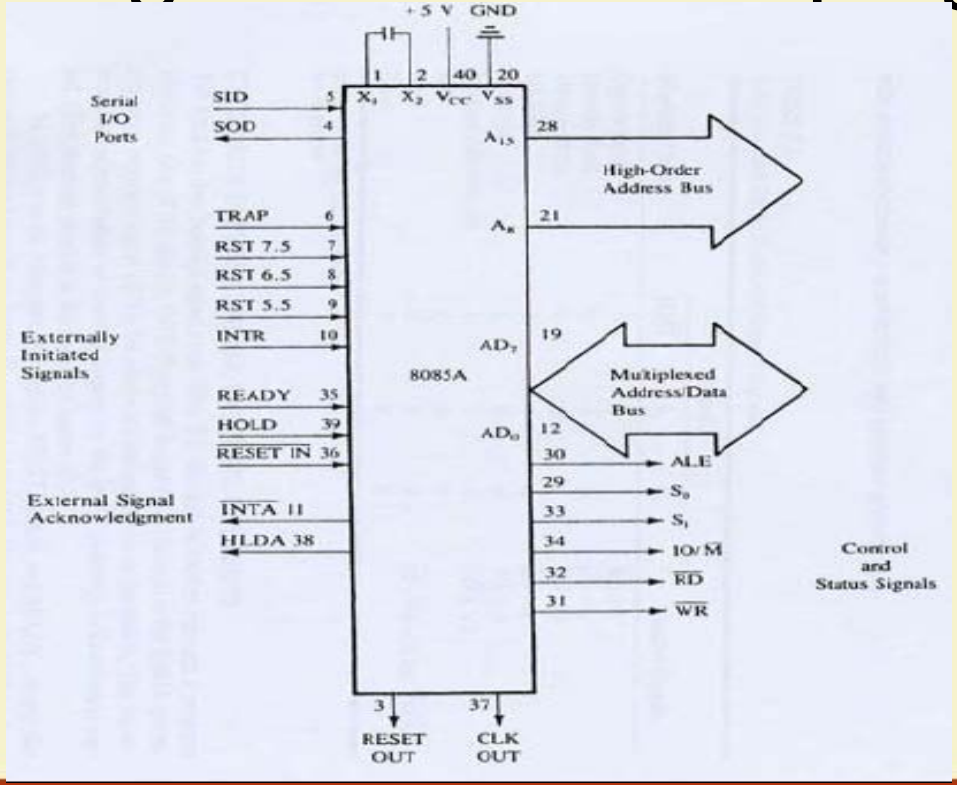
8085 Microprocessor Architecture

- 8-bit general purpose µp
- Capable of addressing 64 k of memory
- Has 40 pins
- Requires +5 v power supply
- Can operate with 3 MHz clock
- 8085 upward compatible

8085 Pin Diagram



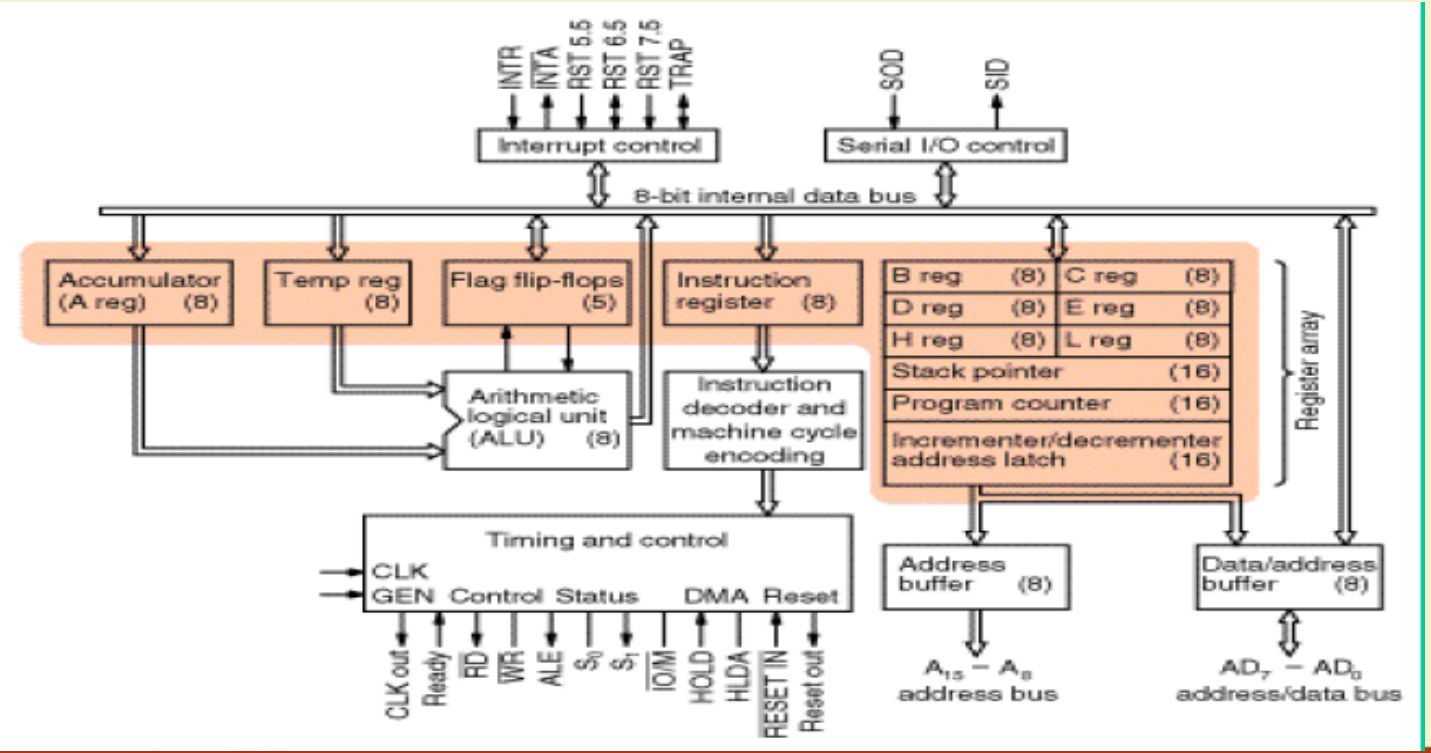
Pin Diagram with Grouping



System Bus

- System Bus – wires connecting memory & I/O to microprocessor
  - Address Bus
    - Unidirectional
    - Identifying peripheral or memory location
  - Data Bus
    - Bidirectional
    - Transferring data
  - Control Bus
    - Synchronization signals
    - Timing signals
    - Control signal

Architecture of Intel 8085 Microprocessor



Intel 8085 Microprocessor

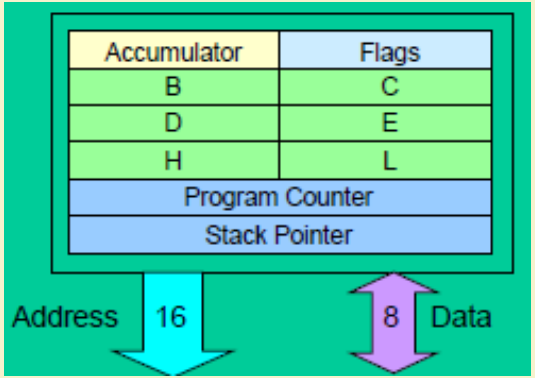
- Microprocessor consists of:
  - **Control unit**: control microprocessor operations.
  - **ALU**: performs data processing function.
  - **Registers**: provide storage internal to CPU.
  - **Interrupts**
  - **Internal data bus**

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.

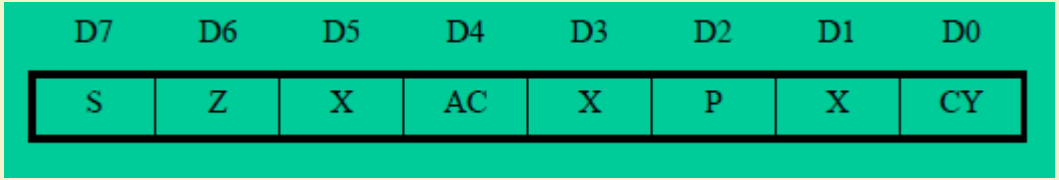
Registers

- General Purpose Registers
  - B, C, D, E, H & L (8 bit registers)
  - Can be used singly
  - Or can be used as 16 bit register pairs
    - BC, DE, HL
  - H & L can be used as a data pointer (holds memory address)
- Special Purpose Registers
  - Accumulator (8 bit register)
    - Store 8 bit data
    - Store the result of an operation



Flag Register

- 8 bit register – shows the status of the microprocessor before/after an operation
- S (sign flag), Z (zero flag), AC (auxillary carry flag), P (parity flag) & CY (carry flag)



Sign Flag

- Used for indicating the sign of the data in the accumulator
- The sign flag is set if negative (1 – negative)
- The sign flag is reset if positive (0 –positive)

Zero Flag

- Is set if result obtained after an operation is 0
- Is set following an increment or decrement operation of that register

Carry Flag

- Is set if there is a carry or borrow from arithmetic operation

Auxillary Carry and Parity

- Auxillary Carry Flag is set if there is a carry out of bit 3
- Parity Flag Is set if parity is even and is cleared if parity is odd

The Internal Architecture

- We have already discussed the general purpose registers, the Accumulator, and the flags.
- 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 Internal Architecture

- 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.

Non Programmable Registers

- Instruction Register & Decoder
  - Instruction is stored in IR after fetched by processor
  - Decoder decodes instruction in IR
- Internal Clock generator
  - 3.125 MHz internally
  - 6.25 MHz externally

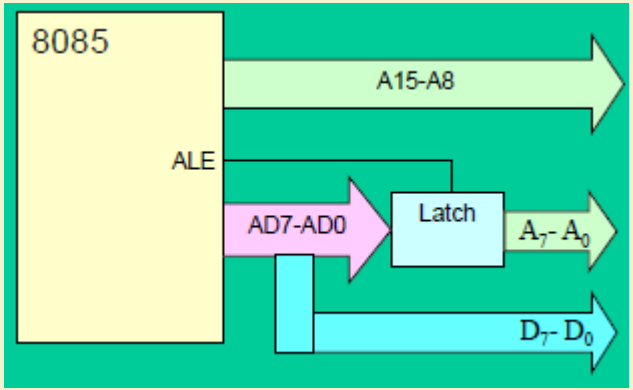
The Address and Data Busses

- 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.

Demultiplexing AD7-AD0

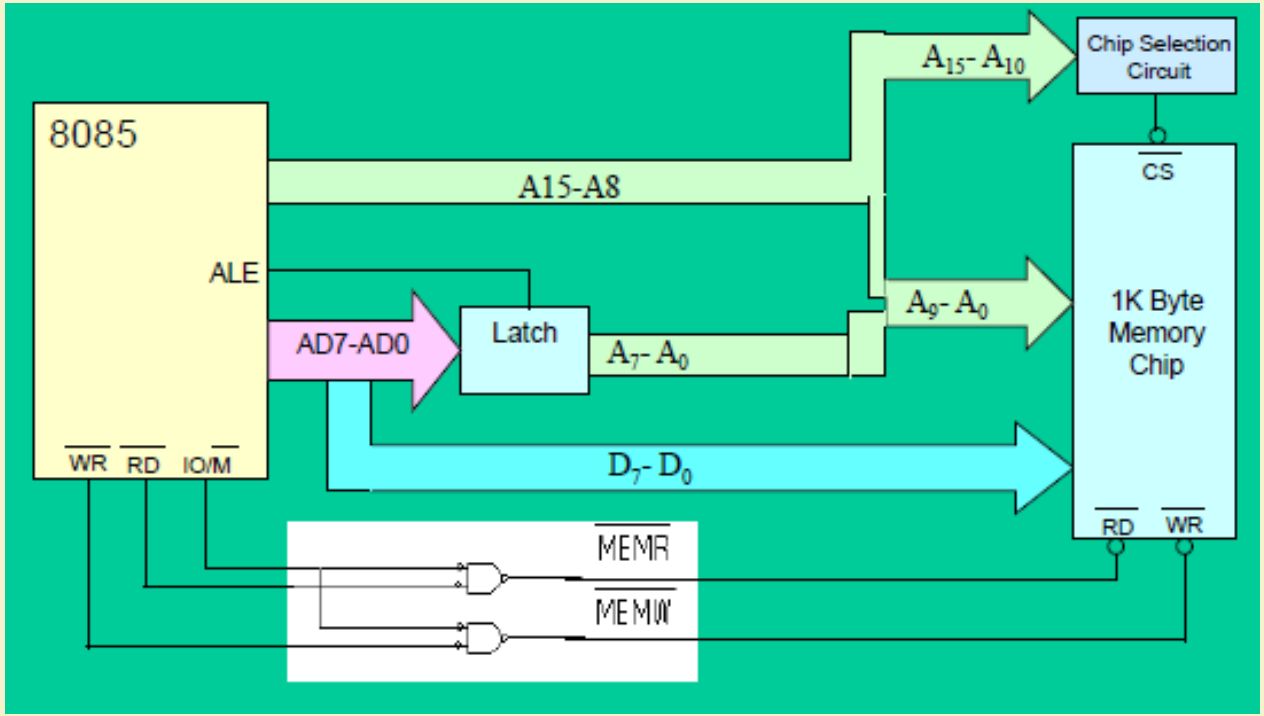
- The high order bits of the address remain on the bus for three clock periods. However, the low order bits remain for only one clock period and they would be lost if they are not saved externally.
- To make sure we have the entire address for the full three clock cycles, we will use an external latch to save the value of AD7– AD0 when it is carrying the address bits. We use the ALE signal to enable this latch.

Demultiplexing AD7-AD0



- Given that ALE operates as a pulse during T1, we will be able to latch the address. Then when ALE goes low, the address is saved and the AD7– AD0 lines can be used for their purpose as the bi-directional data lines.

The Overall Picture



The 8085 Instructions

- Since the 8085 is an 8-bit device it can have up to 2^8 instructions.
  - However, the 8085 only uses 246 combinations that represent a total of 74 instructions.
  - Most of the instructions have more than one format.
- These instructions can be grouped into five different groups:
  - Data Transfer Operations
  - Arithmetic Operations
  - Logic Operations
  - Branch Operations
  - Machine Control Operations

Instruction and Data Formats

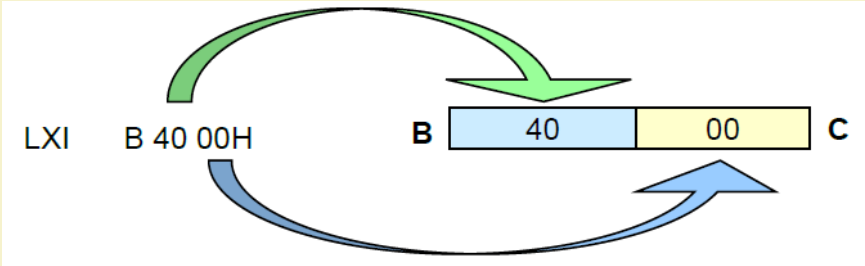
- Each instruction has two parts.
  - The first part is the task or operation to be performed.
    - This part is called the "opcode" (operation code).
  - The second part is the data to be operated on
    - Called the "operand".

Data Transfer Operations

- These operations simply **COPY** the data from the source to the destination.
- MOV, MVI, LDA, and STA
- They transfer:
  - Data between registers.
  - Data Byte to a register or memory location.
  - Data between a memory location and a register.
  - Data between an I\O Device and the accumulator.
- The data in the source is not changed.

The LXI instruction

- The 8085 provides an instruction to place the 16-bit data into the register pair in one step.
  - **LXI Rp, <16-bit address>** (Load eXtended Immediate)
  - The instruction **LXI B 4000H** will place the 16-bit number 4000 into the register pair B, C.
  - The upper two digits are placed in the 1st register of the pair and the lower two digits in the 2nd



The Memory "Register"

- Most of the instructions of the 8085 can use a memory location in place of a register.
  - The memory location will become the "memory" register M.
    - **MOV M B**
    - copy the data from register B into a memory location.
  - Which memory location?
- The memory location is identified by the contents of the HL register pair.
  - The 16-bit contents of the HL register pair are treated as a 16-bit address and used to identify the memory location.

Using the Other Register Pairs

- There is also an instruction for moving data from memory to the accumulator without disturbing the contents of the H and L register.
  - **LDAX Rp** (LoaD Accumulator eXtended)
    - Copy the 8-bit contents of the memory location identified by the Rp register pair into the Accumulator.
    - This instruction only uses the **BC** or **DE** pair.
    - It does not accept the **HL** pair.

Indirect Addressing Mode

- Using data in memory directly (without loading first into a Microprocessor's register) is called Indirect Addressing.

- Indirect addressing uses the data in a register pair as a 16-bit address to identify the memory location being accessed.
  - The HL register pair is always used in conjunction with the memory register "M".
  - The BC and DE register pairs can be used to load data into the Accumultor using indirect addressing.

Arithmetic Operations

- Addition (ADD, ADI):
  - Any 8-bit number.
  - The contents of a register.
  - The contents of a memory location.
  - Can be added to the contents of the accumulator and the result is stored in the accumulator.
- Subtraction (SUB, SUI):
  - Any 8-bit number
  - The contents of a register
  - The contents of a memory location
  - Can be subtracted **from** the contents of the accumulator. The result is stored in the accumulator.

Arithmetic Operations Related to Memory

- These instructions perform an arithmetic operation using the contents of a memory location while they are still in memory.
  - ADD M
    - Add the contents of M to the Accumulator
  - SUB M
    - Sub the contents of M from the Accumulator
  - INR M / DCR M
    - Increment/decrement the contents of the memory location in place.
  - All of these use the contents of the HL register pair to identify the memory location being used.

Arithmetic Operations

– Increment (INR) and Decrement (DCR):
  - The 8-bit contents of any memory location or any register can be directly incremented or decremented by 1.
  - No need to disturb the contents of the accumulator.

Manipulating Addresses

- Now that we have a 16-bit address in a register pair, how do we manipulate it?
  - It is possible to manipulate a 16-bit address stored in a register pair as one entity using some special instructions.

- **INX Rp**
- **DCX Rp**

The register pair is incremented or decremented as one entity. No need to worry about a carry from the lower 8-bits to the upper. It is taken care of automatically.

Logic Operations

- These instructions perform logic operations on the contents of the accumulator.
  - ANA, ANI, ORA, ORI, XRA and XRI
  - Source: Accumulator and
    - An 8-bit number
    - The contents of a register
    - The contents of a memory
  - Destination: Accumulator

ANA	R/M	AND Accumulator With Reg/Mem
ANI	#	AND Accumulator With an 8-bit number
ORA	R/M	OR Accumulator With Reg/Mem
ORI	#	OR Accumulator With an 8-bit number
XRA	R/M	XOR Accumulator With Reg/Mem
XRI	#	XOR Accumulator With an 8-bit number

Logic Operations

– Complement:

- 1’s complement of the contents of the accumulator.
  - CMA No operand

Additional Logic Operations

- Rotate
  - Rotate the contents of the accumulator one position to the left or right.

–	RLC	Rotate the accumulator left. Bit 7 goes to bit 0 AND the Carry flag.
–	RAL	Rotate the accumulator left through the carry. Bit 7 goes to the carry and carry goes to bit 0.
–	RRC	Rotate the accumulator right. Bit 0 goes to bit 7 AND the Carry flag.
–	RAR	Rotate the accumulator right through the carry. Bit 0 goes to the carry and carry goes to bit 7.