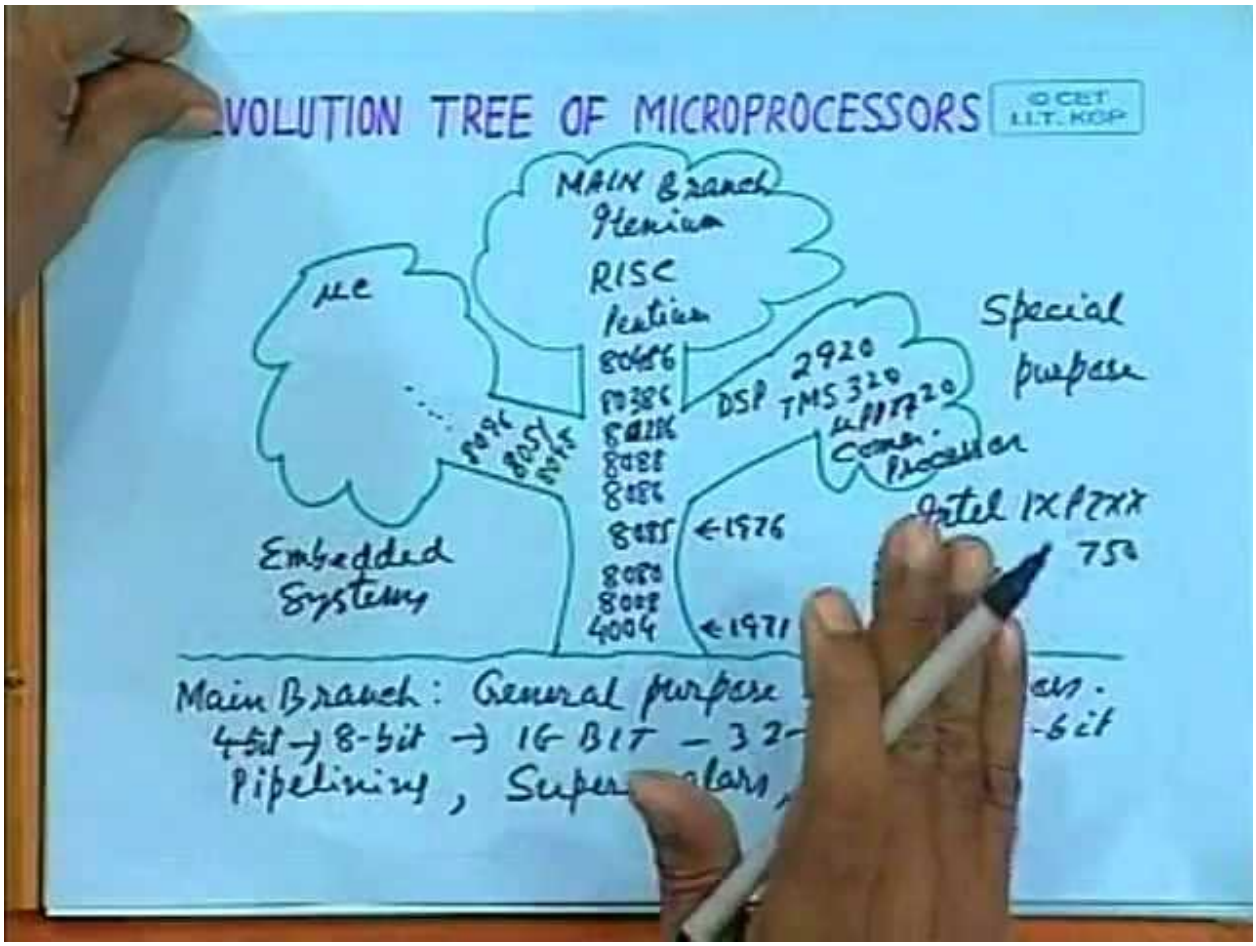
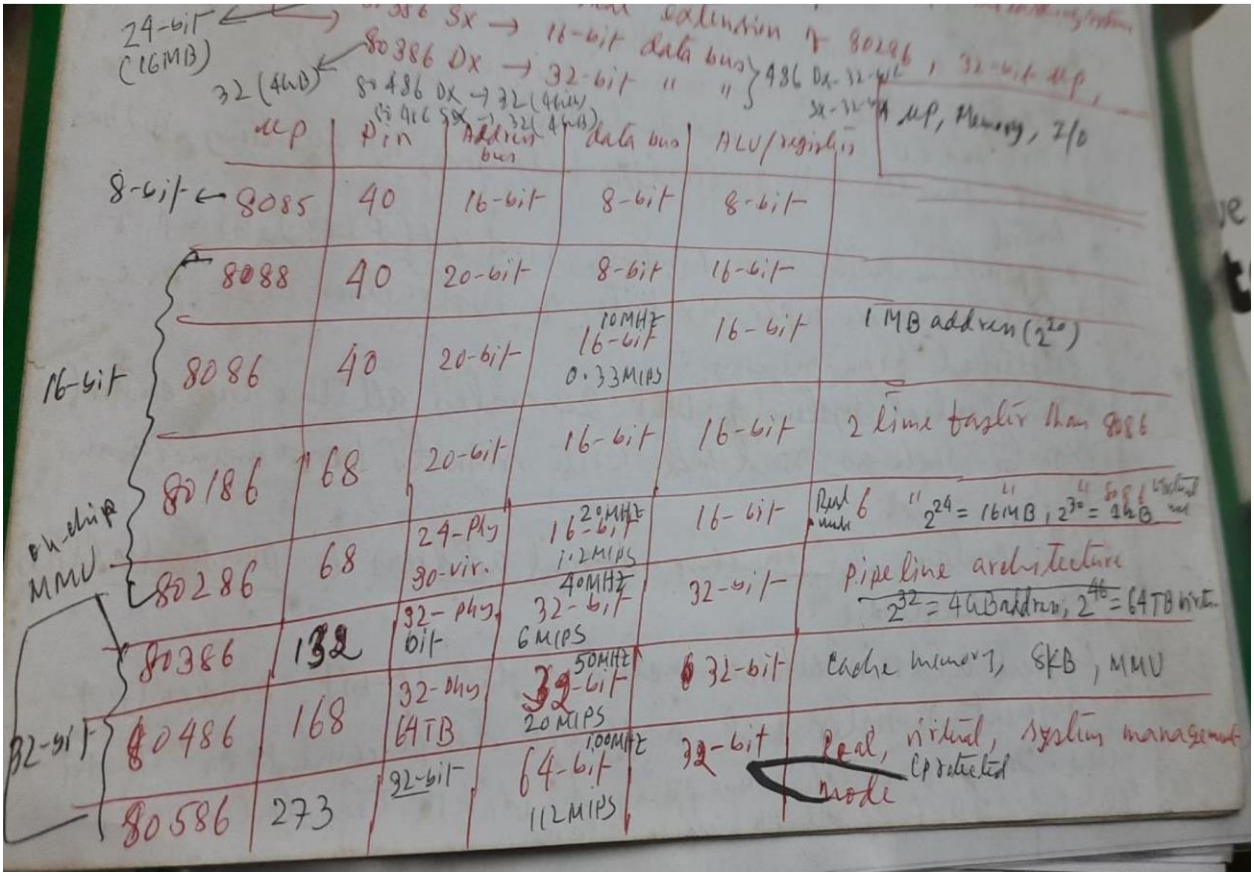
Microprocessor Online Lecture 1:

Evolution Tree of Microprocessor:

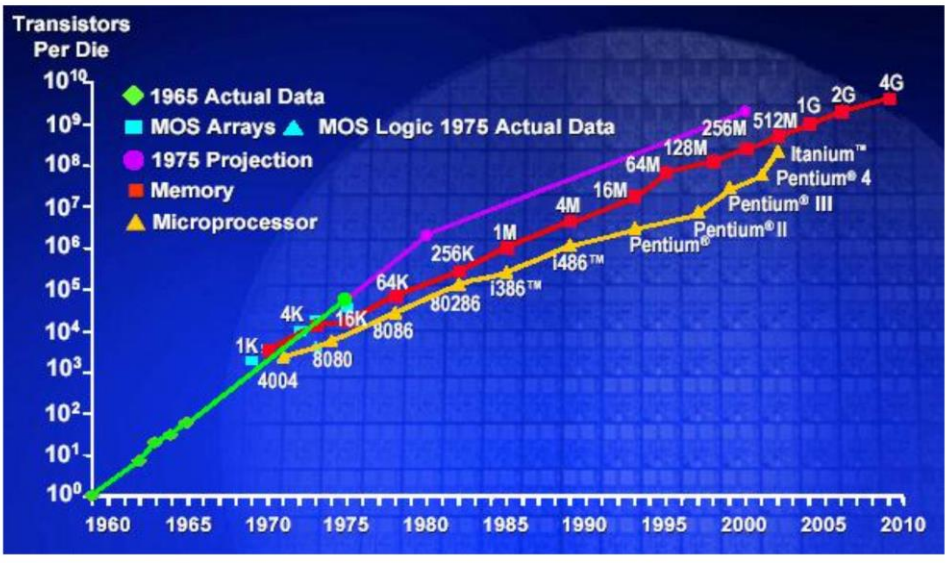


NAME	YEAR	TRANSISTORS	DATA WIDTH	CLOCK SPEED
8080	1974	6,000	8 bits	2 MHz
8085	1976	6,500	8 bits	5 MHz
8086	1978	29,000	16 bits	5 MHz
8088	1979	29,000	8 bits	5 MHz
80286	1982	134,000	16 bits	6 MHz
80386	1985	275,000	32 bits	16 MHz
80486	1989	1,200,000	32 bits	25 MHz
PENTIUM	1993	3,100,000	32/64 bits	60 MHz
PENTIUM II	1997	7,500,000	64 bits	233 MHz
PENTIUM III	1999	9,500,000	64 bits	450 MHz
PENTIUM IV	2000	42,000,000	64 bits	1.5 GHz



Moore's Law

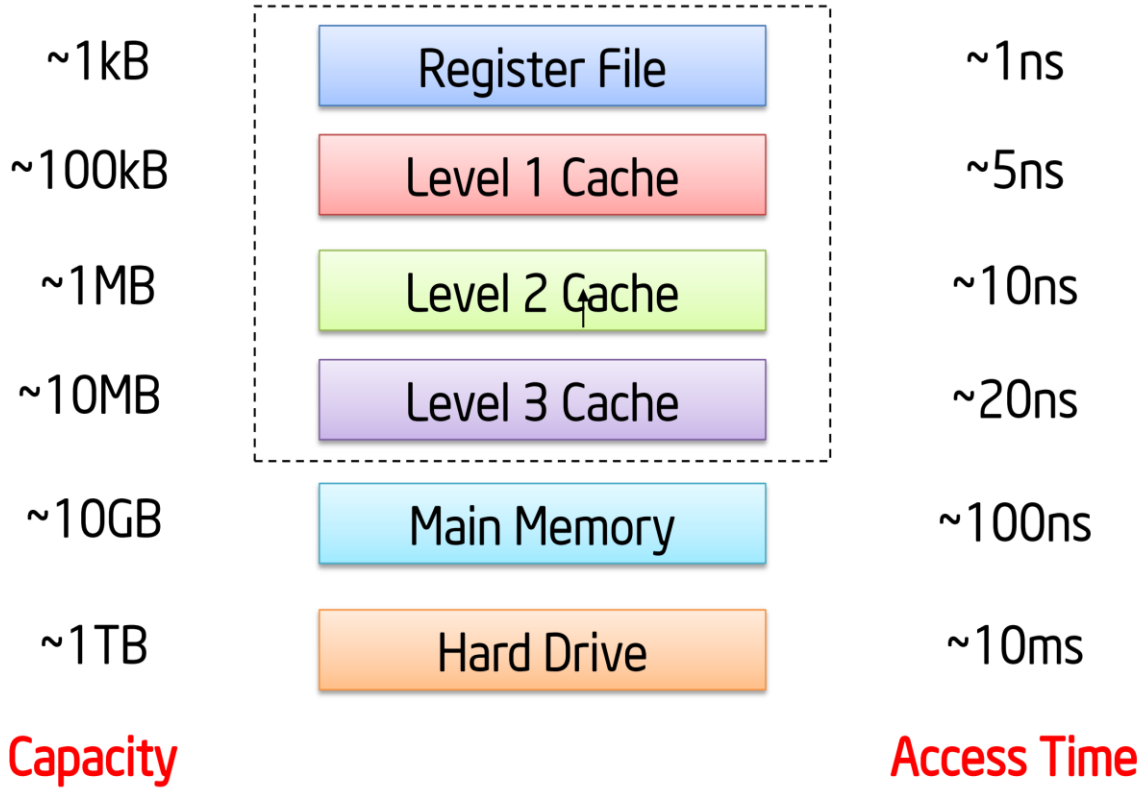
- "The number of transistors incorporated in a chip will approximately double every 24 months." (1965)



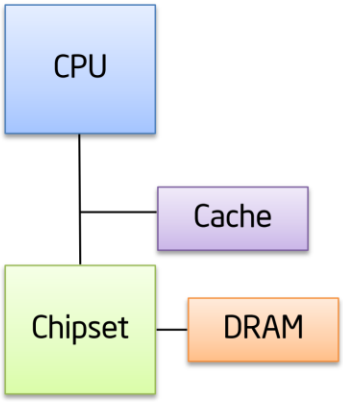
PC Components

- Microprocessor — performs all computations
- Cache — fast memory which holds current data and program
- Main memory — larger DRAM memory contains more data
- Chipset — controls communication between components
- Motherboard — circuit board which holds all the above components
- Peripheral cards — controls added computer accessories

Memory Hierarchy

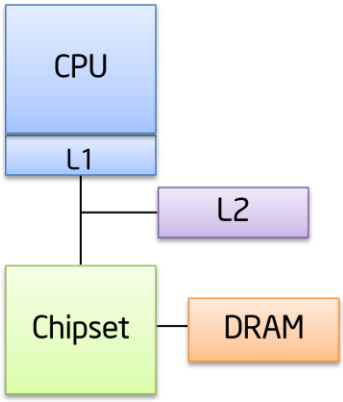


Memory Hierarchy Evolution



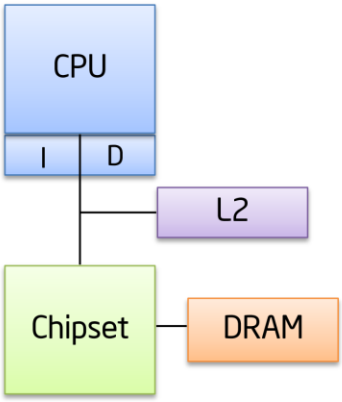
386

No on-die cache.
Level 1 cache
on motherboard



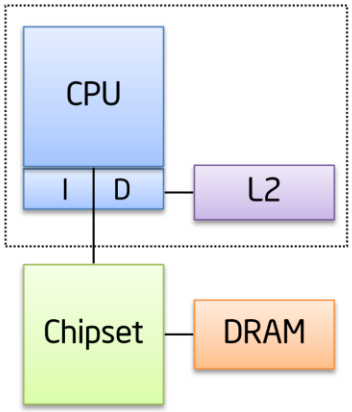
486

Level 1 cache on-die.
Level 2 cache
on motherboard



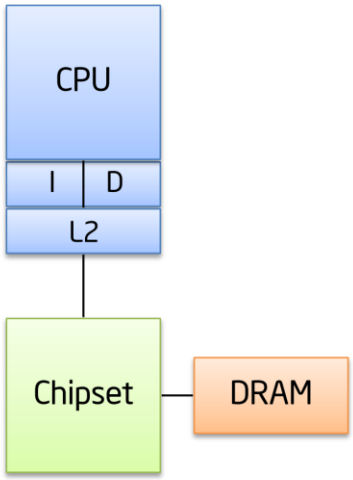
Pentium

Separate Instruction
and Data Caches



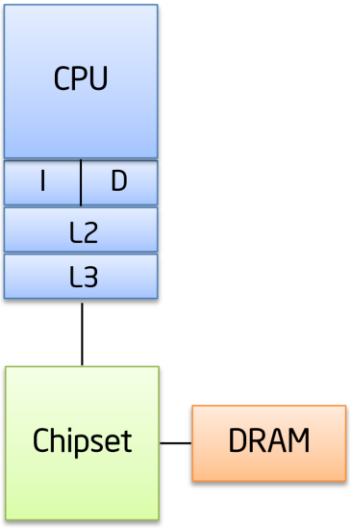
Pentium II

Separate bus to L2
cache in same
package



Pentium III

L2 cache on-die



Core i7

L3 cache on-die

What is Architecture?

- Computer architecture is defined by the instructions a processor can execute
- Programs written for one processor can run on any other processor of the same architecture

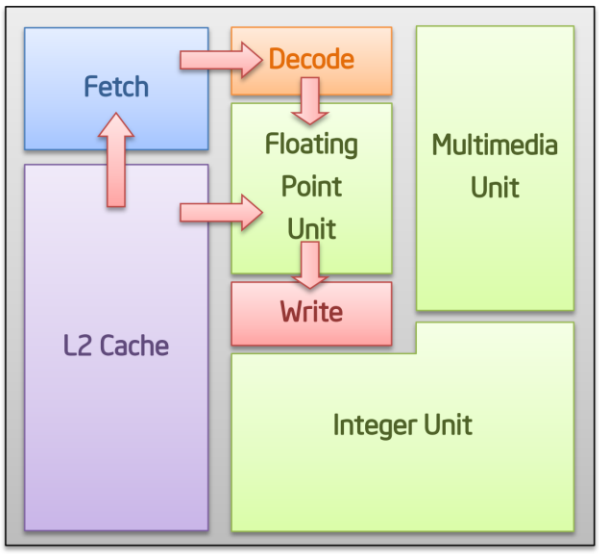
What are Instructions?

- Instructions are the most basic actions the processor can take:
  - ADD AX, BX — Add value AX to BX and store in AX
  - CMP AX, 5 — Compare value in AX to 5
  - JE 16 — Jump ahead 16 bytes if comparison was equal

What is Microarchitecture?

- Microarchitecture is the steps a processor takes to execute a particular set of instructions
- Processors of the same architecture have the same instructions but may carry them out in different ways
- Microarchitecture Features:
  - Cache memory
  - Pipelining
  - Out-of-Order Execution
  - Superscalar Issue

Simplified Microprocessor



Fetch Unit gets the next instruction from the cache.

Decode Unit determines type of instruction.

Instruction and data sent to Execution Unit.

Write Unit stores result.

Sequential Processing (386)

Cycle	1	2	3	4	5	6	7	8	9
$Instr_1$	Fetch	Decode	Execute	Write					
$Instr_2$					Fetch	Decode	Execute	Write	
$Instr_3$									Fetch

- Sequential processing works on one instruction at a time

Pipelined Processing (486)

Cycle	1	2	3	4	5	6	7	8	9
$Instr_1$	Fetch	Decode	Execute	Write					
$Instr_2$		Fetch	Decode	Execute	Write				
$Instr_3$			Fetch	Decode	Execute	Write			
$Instr_4$				Fetch	Decode	Execute	Write		
$Instr_5$					Fetch	Decode	Execute	Write	
$Instr_6$						Fetch	Decode	Execute	Write

- Latency — elapsed time from start to completion of a particular task
- Throughput — how many tasks can be completed per unit of time
- Pipelining only improves throughput
  - Each job still takes 4 cycles to complete

In-Order Pipeline (486)

Cycle	1	2	3	4	5	6	7	8	9
$Instr_1$	Fetch	Decode	Execute			Write			
$Instr_2$		Fetch	Decode	Wait		Execute	Write		
$Instr_3$			Fetch	Decode	Wait		Execute	Write	
$Instr_4$				Fetch	Decode	Wait		Execute	Write
$Instr_5$					Fetch	Decode	Wait		Execute
$Instr_6$						Fetch	Decode	Wait	

- In-Order execution requires instructions to be executed in the original program order

Out-of-Order Execution (Pentium II)

Cycle	1	2	3	4	5	6	7	8	9
$Instr_1$	Fetch	Decode	Execute			Write			
$Instr_2$		Fetch	Decode	Wait		Execute	Write		
$Instr_3$			Fetch	Decode	Execute	Write			
$Instr_4$				Fetch	Decode	Wait	Execute	Write	
$Instr_5$					Fetch	Decode	Execute	Write	
$Instr_6$						Fetch	Decode	Execute	Write

- Program Order vs Dataflow Order
- Dataflow: data-driven scheduling of events
  - The start of an event should be enabled by the availability of its required input (data dependency)

Superscalar Issue (Pentium)

Cycle	1	2	3	4	5	6	7	8	9
$Instr_1$	Fetch	Decode	Execute			Write			
$Instr_2$	Fetch	Decode	Wait			Execute	Write		
$Instr_3$		Fetch	Decode	Execute	Write				
$Instr_4$		Fetch	Decode	Wait			Execute	Write	
$Instr_5$			Fetch	Decode	Execute	Write			
$Instr_6$			Fetch	Decode	Execute	Write			
$Instr_7$				Fetch	Decode	Execute	Write		
$Instr_8$				Fetch	Decode	Execute	Write		

- Superscalar issue allows multiple instructions to be issued at the same time

Microarchitecture and Performance

- Performance is measured by how long a processor takes to run a program
- Time is reduced by increasing Instructions Per Cycle (IPC) and clock rate
- Microarchitecture affects IPC and clock rate:
  - More pipe stages
    - Less work in each cycle means better clock rate
    - More dependencies means worse IPC
  - Superscalar Issue and Out-of-Order Execution
    - Parallel work means better IPC
    - More complexity can mean worse clock rate

Measuring Processor Performance

- Clock Rate
  - Simplest but not especially accurate
- Instructions Per Cycle (IPC)
  - Not meaningful without clock rate
  - Varies from program to program
- SPEC Performance
  - Standard Performance Evaluation Corporation
  - Tests PCs using benchmark suite of programs
    - SPECint: Integer intensive programs
    - SPECfp: Floating point intensive programs
- TPC Performance
  - Transaction Performance Council
  - Tests servers and workstations

8085 MICROPROCESSOR ARCHITECTURE

The 8085 microprocessor is an 8-bit processor available as a 40-pin IC package and uses +5 V for power. It can run at a maximum frequency of 3 MHz. Its data bus width is 8-bit and address bus width is 16-bit, thus it can address 2^{16} = 64 KB of memory.

Arithmetic and Logic Unit

The ALU performs the actual numerical and logical operations such as Addition (ADD), Subtraction (SUB), AND, OR etc. It uses data from memory and from Accumulator to perform operations. The results of the arithmetic and logical operations are stored in the accumulator.

Registers

The 8085 includes six registers, one accumulator and one flag register, as shown in Fig. 3. In addition, it has two 16-bit registers: stack pointer and program counter. They are briefly described as follows.

The 8085 has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H and L. they can be combined as register pairs - BC, DE and HL to perform some 16-bit operations. The programmer can use these registers to store or copy data into the register by using data copy instructions.

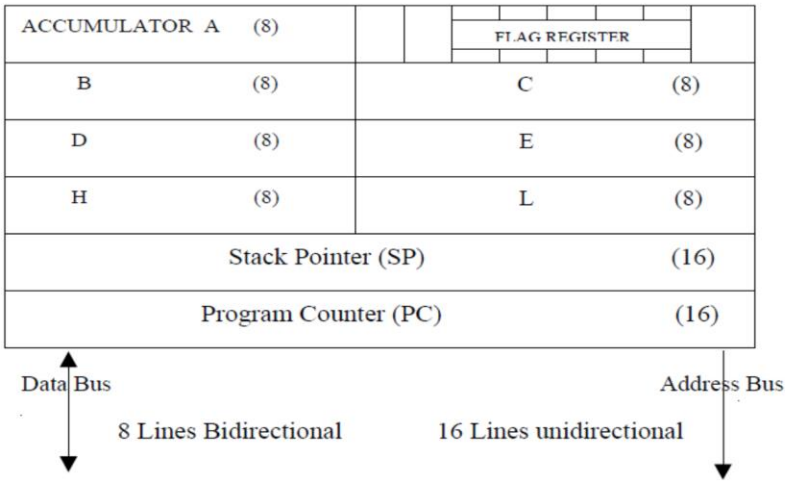


Fig. 3 Register organisation

Accumulator

The accumulator is an 8-bit register that is a part of ALU. This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

Flag register

The ALU includes five flip-flops, which are set or reset after an operation according to data condition of the result in the accumulator and other registers. They are called Zero (Z), Carry (CY), Sign (S), Parity (P) and Auxiliary Carry (AC) flags. Their bit positions in the flag register are shown in Fig. 4. The microprocessor uses these flags to test data conditions.

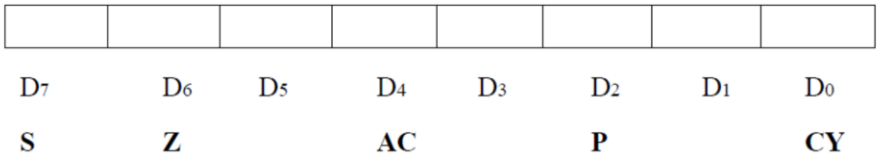


Fig. 4 Flag register

Program Counter (PC)

This 16-bit register deals with sequencing the execution of instructions. This register is a memory pointer. The microprocessor uses this register to sequence the execution of the instructions. The function of the program counter is to point to the memory address from which the next byte is to be fetched. When a byte is being fetched, the program counter is automatically incremented by one to point to the next memory location.

Stack Pointer (SP)

The stack pointer is also a 16-bit register, used as a memory pointer. It points to a memory location in R/W memory, called stack. The beginning of the stack is defined by loading 16-bit address in the stack pointer.

Instruction Register/Decoder

It is an 8-bit register that temporarily stores the current instruction of a program. Latest instruction sent here from memory prior to execution. Decoder then takes instruction and decodes or interprets the instruction. Decoded instruction then passed to next stage.

The schematic representation of the 8085 bus structure is as shown in Fig. 5. The microprocessor performs primarily four operations:

I. Memory Read: Reads data (or instruction) from memory.
II. Memory Write: Writes data (or instruction) into memory.
III. I/O Read: Accepts data from input device.
IV. I/O Write: Sends data to output device.

The 8085 processor performs these functions using address bus, data bus and control bus as shown in Fig. 5.

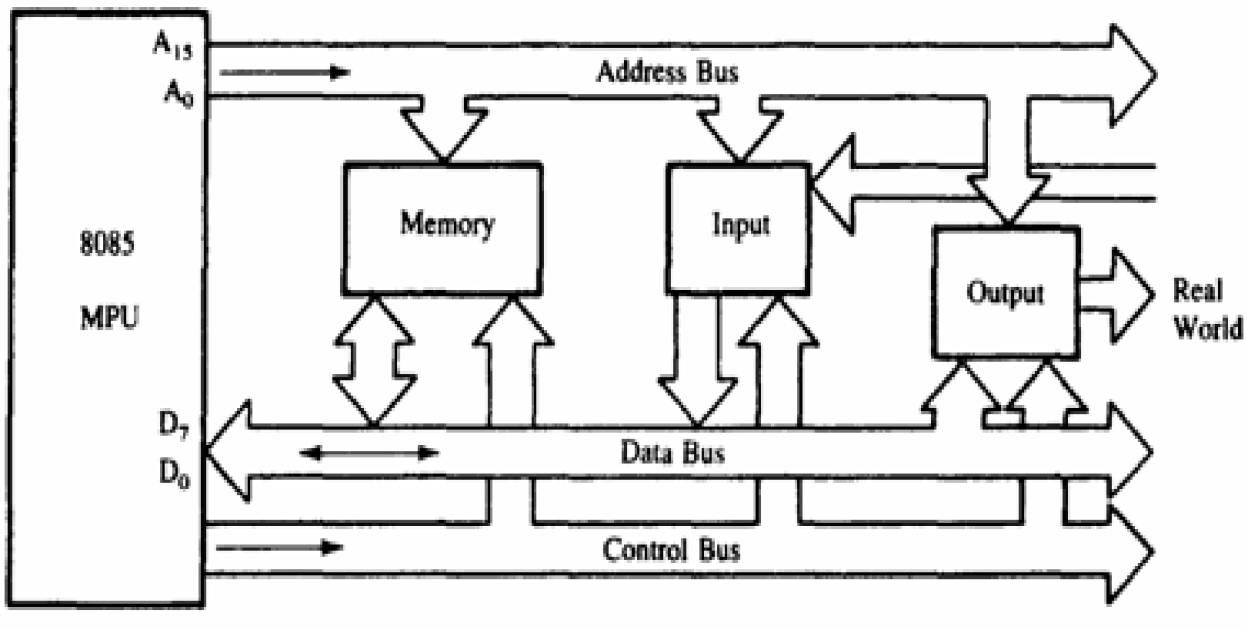


Fig. 5 The 8085 bus structure

INSTRUCTION SET AND EXECUTION IN 8085

Based on the design of the ALU and decoding unit, the microprocessor manufacturer provides instruction set for every microprocessor. The instruction set consists of both machine code and mnemonics.

An instruction is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions that a microprocessor supports is called instruction set. Microprocessor instructions can be classified based on the parameters such functionality, length and operand addressing.

Classification based on functionality:

I. Data transfer operations: This group of instructions copies data from source to destination. The content of the source is not altered.
II. Arithmetic operations: Instructions of this group perform operations like addition, subtraction, increment & decrement. One of the data used in arithmetic operation is stored in accumulator and the result is also stored in accumulator.
III. Logical operations: Logical operations include AND, OR, EXOR, NOT. The operations like AND, OR and EXOR uses two operands, one is stored in accumulator and other can be any register or memory location. The result is stored in accumulator. NOT operation requires single operand, which is stored in accumulator.
IV. Branching operations: Instructions in this group can be used to transfer program sequence from one memory location to another either conditionally or unconditionally.
V. Machine control operations: Instruction in this group control execution of other instructions and control operations like interrupt, halt etc.

Classification based on length:

I. One-byte instructions: Instruction having one byte in machine code. Examples are depicted in Table 2.
I. Two-byte instructions: Instruction having two byte in machine code. Examples are depicted in Table 3
II. Three-byte instructions: Instruction having three byte in machine code. Examples are depicted in Table 4.

Table 2 Examples of one byte instructions

Opcode	Operand	Machine code/Hex code
MOV	A, B	78
ADD	M	86

Table 3 Examples of two byte instructions

Opcode	Operand	Machine code/Hex code	Byte description
MVI	A, 7FH	3E	First byte
		7F	Second byte
ADI	0FH	C6	First byte
		0F	Second byte

Table 4 Examples of three byte instructions

Opcode	Operand	Machine code/Hex code	Byte description
JMP	9050H	C3	First byte
		50	Second byte
		90	Third byte
LDA	8850H	3A	First byte
		50	Second byte
		88	Third byte

In **computer** science, an **instruction** is a single operation of a processor defined by the processor **instruction** set. The size or length of an **instruction** varies widely, from as little as 4-bits in some microcontrollers to many as multiples of a bytes in some very long **instruction** word (VLIW) systems.

Fetch **instruction** from memory. Decode the **instruction**. Read the effective address from memory. Execute the **instruction**.

An **instruction format** defines layout of bits of an **instruction**, in terms of its constituent parts. An **instruction format** must include an opcode and implicitly or explicitly, zero or more operands. Each explicit operand is referenced using one of addressing modes.

General Instruction Format

Opcode-Field	**Operand/Address-Field**

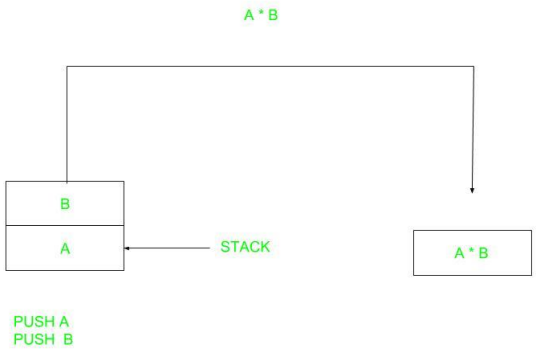
Op-field: specifies the operation to be performed;
Address-field: provides operands or the CPU register/MM addresses of the operands.

In computing, an **operand** is the part of a computer instruction, which specifies what data is to be manipulated or operated on, while at the same time representing the data itself. ... Depending on the instruction, there may be zero, one, two, or more **operands**.

Types of Operands

- Addresses.
- Numbers.
- Characters.
- Logical data.

Zero Address Instructions –



- A stack based computer do not use address field in instruction.To evaluate a expression first it is converted to revere Polish Notation i.e. Post fix Notation.
- `Expression: X = (A+B)*(C+D)`
- `Post fixed : X = AB+CD+*`
- `TOP means top of stack`
- `M[X] is any memory location`

PUSH	A	TOP = A
PUSH	B	TOP = B
ADD		TOP = A+B
PUSH	C	TOP = C
PUSH	D	TOP = D
ADD		TOP = C+D
MUL		TOP = (C+D)*(A+B)
POP	X	M[X] = TOP

One Address Instructions –
This use a implied ACCUMULATOR register for data manipulation. One operand is in accumulator and other is in register or memory location. Implied means that the CPU already know that one operand is in accumulator so there is no need to specify it.

- `Expression: X = (A+B)*(C+D)`
- `AC is accumulator`
- `M[] is any memory location`
- `M[T] is temporary location`

LOAD	A	AC = M[A]
ADD	B	AC = AC + M[B]
STORE	T	M[T] = AC
LOAD	C	AC = M[C]
ADD	D	AC = AC + M[D]
MUL	T	AC = AC * M[T]
STORE	X	M[X] = AC

Two Address Instructions –
This is common in commercial computers. Here two address can be specified in the instruction. Unlike earlier in one address instruction the result was stored in accumulator here result cab be stored at different location rather than just accumulator, but require more number of bit to represent address.

Here destination address can also contain operand.

```
Expression: X = (A+B)*(C+D)
R1, R2 are registers
M[] is any memory location
```

MOV	R1, A	R1 = M[A]

ADD	R1, B	R1 = R1 + M[B]
MOV	R2, C	R2 = C
ADD	R2, D	R2 = R2 + D
MUL	R1, R2	R1 = R1 * R2
MOV	X, R1	M[X] = R1

Three Address Instructions –

This has three address field to specify a register or a memory location. Program created are much short in size but number of bits per instruction increase. These instructions make creation of program much easier but it does not mean that program will run much faster because now instruction only contain more information but each micro operation (changing content of register, loading address in address bus etc.) will be performed in one cycle only.

- `Expression: X = (A+B)*(C+D)`
- `R1, R2 are registers`
- `M[] is any memory location`

ADD	R1, A, B	R1 = M[A] + M[B]
ADD	R2, C, D	R2 = M[C] + M[D]
MUL	X, R1, R2	M[X] = R1 * R2

Instruction Format

Instruction		Comment
SUB	Y, A, B	Y ← A − B
MPY	T, D, E	T ← D × E
ADD	T, T, C	T ← T + C
DIV	Y, Y, T	Y ← Y ÷ T

(a) Three-address instructions

Instruction		Comment
MOVE	Y, A	Y ← A
SUB	Y, B	Y ← Y − B
MOVE	T, D	T ← D
MPY	T, E	T ← T × E
ADD	T, C	T ← T + C
DIV	Y, T	Y ← Y ÷ T

(b) Two-address instructions

Instruction		Comment
LOAD	D	AC ← D
MPY	E	AC ← AC × E
ADD	C	AC ← AC + C
STOR	Y	Y ← AC
LOAD	A	AC ← A
SUB	B	AC ← AC − B
DIV	Y	AC ← AC ÷ Y
STOR	Y	Y ← AC

(c) One-address instructions

Figure 10.3 Programs to Execute $Y = \frac{A - B}{C + (D \times E)}$

Lecture Prepared by

Md. Javed Hossain
Associate Professor
CSTE, NSTU.