8086 Microprocessor

Course Title: Computer Organization and Architecture

Dr. Nazib Abdun Nasir
Assistant Professor
CS, AIUB
nazib.nasir@aiub.edu

Lecture Outline

1. A brief survey of 8086 family
2. The architecture of the 8086
3. The registers and their special functions
4. Overall structure of the IBM PC
5. The memory organization
6. I/O ports, DOS and BIOS routines

Intel 8o86 Family of Microprocessors

Based on Intel 8086 family of microprocessor

- IBM PC (8088)
- PC XT (8088)
- PC AT (80286)
- PS/1 (80286)
- PS/2 (8086/ 80286/ 80386/ 80486)
- Some PC compatible Laptop models (80186)

The 8086 and 8088 Microprocessors

<table>
<tr><th>8086</th><th>8088</th></tr>
<tr><td>➢ The first 16-bit Microprocessor that can process 16-bits of data at a time. [1978].8086 has 16-bit data bus</td><td>➢ Internally 8088 is essentially same as 8086.however, externally 8088 has a 8-bit data bus</td></tr>
<tr><td>➢ 8086 has faster clock rate and thus has better performance.</td><td>➢ It is less expensive to build the computer around 8088 [used to build IBM original PC]</td></tr>
<tr><td colspan="2">➢ Both 8086 and 8088 have the same instruction set and it forms the basic set of instruction for other microprocessors in the family.</td></tr>
</table>

80186 and 80188 Microprocessors

80186	80188
➢ 80186 is the enhanced version of 8086	➢ 80188 is the enhanced version of 8088
➢ 80816 incorporates with all the functions of the 8086 in addition to some support chips	➢ 80818 incorporates with all the functions of the 8088 in addition to some support chips
➢ Both 80186 and 80188 can execute some extended instruction set. ➢ 80186 and 80188 have no significant advantage over the 8086 and 8088. ➢ 80186 and 80188 were overshadowed very soon due to the development of 80286.	

The 80286 Microprocessor

- **80286** is 16-bit microprocessor and was introduced in 1982.
- It operates **faster** than the 8086 [12.5 MHz vs 10MHz]
- **Two modes of operation**
  - **Real address:**
  - ➢ In this mode, microprocessor behaves like 8086 and programs for 8086 can be executed without modification.
  - **Protected Virtual address:**
    - ➢ **Multitasking:** 80286 supports multi-tasking. So, it can operate several tasks at the same time.
    - ➢ **Memory protection:** Protects the memory used by one program from the actions of another program.

The 80286 Microprocessor

- **More addressable Memory:** The protected mode can address 16 megabytes of memory as opposed to 1MB in 8086 and 8088.

- **Virtual Memory in protected mode:** 80286 can consider external storage or disk as a physical memory and execute large programs [up to 1GB or 2^30 bytes].

The 80386 and 80386SX Microprocessor

80386

- 80386 or 386 is the first 32-bit microprocessor introduced in 1985
- Much faster than 80286 as it has 32-bit data path, high clock rate (up to 33MHz) and the ability to execute instruction in fewer clock cycles than 80286 .
- Like 80286, the 386 can operate in two operational mode (real or protected).
- In real mode, 386 behaves like and 8086, however, it can emulate the 80286 in protected mode
- The 386 also has a virtual 8086 mode that is designed to run multiple 8086 applications under memory protection.
- The 386 protected mode can address 4 gigabytes of physical memory and 64 terabytes of virtual memory.

80386SX

- The 386SX has essentially the same internal structure as the 386, however, it has only a 16-bit data bus.

The 80486 and 80486SX Microprocessor

80486

- 80486 or 486 is another 32-bit microprocessor introduced in 1989.
- 486 is the fastest and most powerful microprocessor in the family.
- 486 incorporates all the functions of the 386 with some other supportive chips including 80387 numeric processor like 80387.
- 80387 numeric processor performs floating-point number operations.
- An 8-KB cache memory serves as a fast memory area to buffer the data coming from a slower memory unit.
- With numeric processor, cache memory and faster design, 486 is three times faster than 386 running at the same clock speed.

80486SX

- The 486SX has essentially the same internal structure as the 486, however, it does not have any floating point processor.

Organization of the 8086/8088 Microprocessor

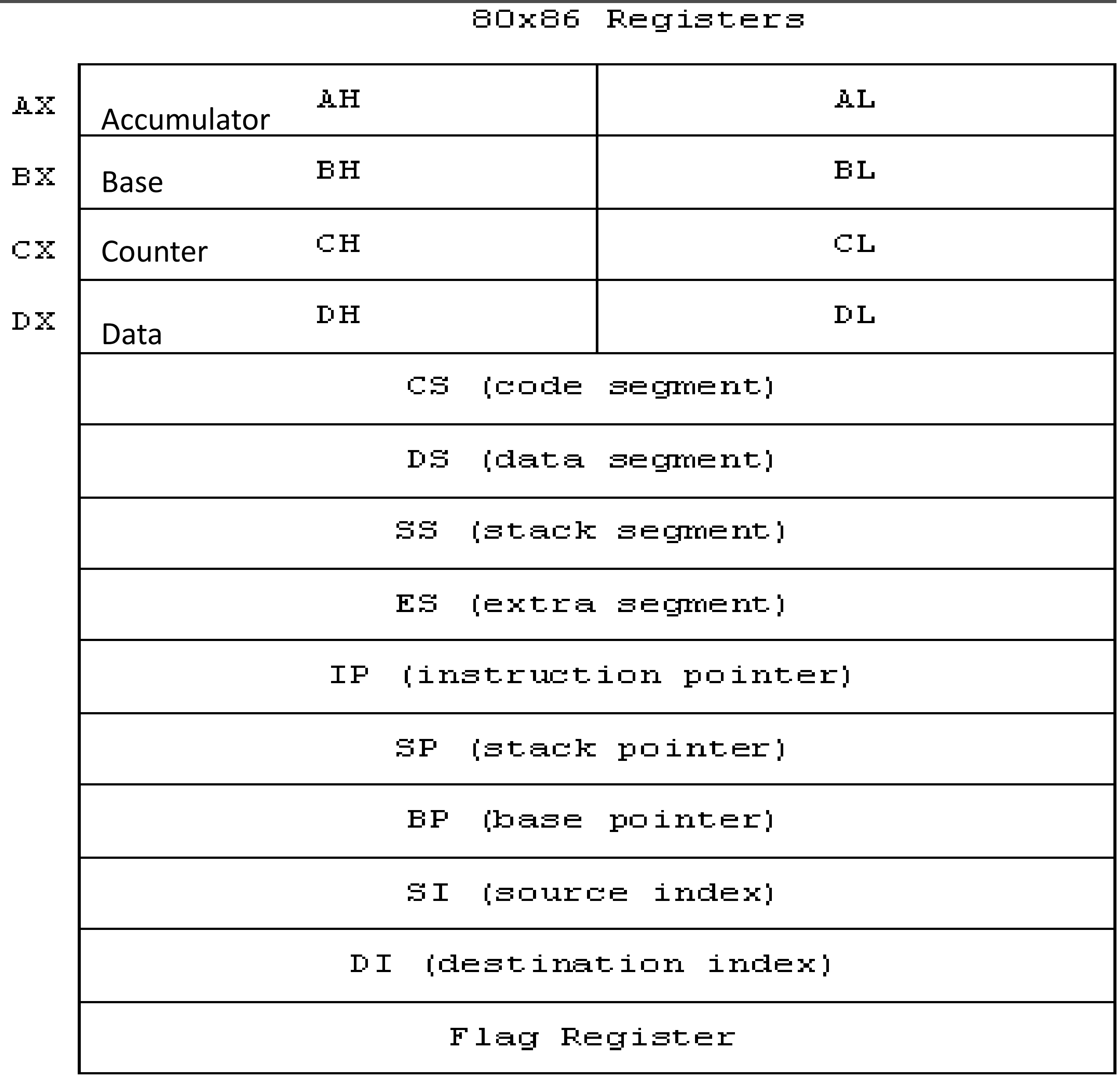
- 8086 and 8088 has the simplest structure and they provide the insights to the most advanced processors.
- As both 8086 and 8088 essentially has the same structure, we will use the term “8086” for both.

Registers

- Information inside the microprocessor is stored in **registers.**
- Registers are classified according to their functions.
    - ➢ Data register holds the data for an operation.
    - ➢ Address register holds the address for an instruction or data.
    - ➢ Status register keeps the current status of the processor.
  - ➢ **Data registers:** 8086 has **four** general data registers.
  - ➢ **Address registers:** Segment, Pointer, and Index registers.
  - ➢ **Status register:** FLAG registers.
- **There are total fourteen 16-bit registers.**

*** The good news is, we DO NOT need to memorize them at all. it will become familiar as we go on. :)

8086 Registers



80x86 Registers

General Register	AX	Accumulator AH	AL
	BX	Base BH	BL
	CX	Counter CH	CL
	DX	Data DH	DL
Segment Register		CS (code segment)	
		DS (data segment)	
		SS (stack segment)	
		ES (extra segment)	
Pointer and Index Register		IP (instruction pointer)	
		SP (stack pointer)	
		BP (base pointer)	
		SI (source index)	
		DI (destination index)	
FLAG Register		Flag Register	

AX: Accumulator Register

- ***** Address registers store addresses, instructions, and data in memory.**

- Use of AX register generates shortest machine code.

- Thus, AX is preferred register to use in **arithmetic, logic,** and **data transfer** instructions.

- In **multiplication and division**, one of the numbers involved must be in AX or AL.

- Input and Output also require the use of AL or AX.

BX: Base Register

- BX also serves as an address register.

  i.e. Table look-up instruction (XLAT)

CX: Counter Register

- CX is used as program’s **loop counter.**
- CX is also used as a counter to control **string operations.**
- CL is used as count in **bit rotation and shifting instructions.**

DX: Data Register

- DX is used in **multiplication, division, and I/O operations.**

Memory Recall

- **List one special function of AX, BX, CX, and DX**

Segment Registers: CS,DS,SS,ES

- Memory is a collection of bytes.
- Each memory bytes has an address starting with 0.
- The 8086 assigns 20-bit physical address to its memory location. [i.e. we can address 2^20(1MB) of memory]. Thus the first byte of the memory addresses:

Binary representation	Hex Representation
0000 0000 0000 0000 0000	00000h
0000 0000 0000 0000 0001	00001h
0000 0000 0000 0000 0010	00010h
0000 0000 0000 0000 0011	00011h
0000 0000 0000 0000 0100	00100h

- ***So what will be the highest address of 20-bit memory address?

Using 20-bit Address in 16-bit Processor

- To explain segment register’s function, lets have a look on the idea of memory segments.

- How can we fit 20 bit address into 16 bit register?

Memory Partitioning Into Segments

- A memory segment is a block of 2^16 (64KB) consecutive memory bytes.

- Each segment is identified by segment number. [starts with 0]

- A **segment** number is 16-bit [thus, highest value FFFFh].

- Within a segment, a memory location is specified by giving an offset.

- **Offset:** Number of bytes from the beginning of segment.

i.e. for a 64KB segment, the offset can be given as 16-bit number.

- ➢ **The first byte in a segment has offset 0000h.**

- ➢ **The Last byte in a segment has offset FFFFh**

Segment : Offset address

- A memory segment may be specified by providing a segment number and an offset.

- Memory segment is written in the form of *segment : offset*.

- The representation of *segment : offset* is known as logical address.

- e.g. A4FB:4872h means offset 4872h within segment A4FBh.

How to obtain 20-bit physical address in a 16-bit microprocessor?

1. The 8086 shifts the segment address 4-bits to the left [i.e. multiply by 10h].

2. Add the offset address to the segment address.

i.e. to get the 20-bit physical address from A4FB:4872h,

A4FB0h [multiplied segment with 10]

+ 4872h

======

A9822h [20-bit physical address]

Task

- Find the 20-bit address of ABC4:12BAh?

Location of Segments

- Segment 0 starts at address 0000:0000h=00000h and ends at 0000:FFFFh=0FFFFh.

- Segment 1 starts at address 0001:0000h=00010h and ends at 0001:FFFFh=1000Fh.

- Segment 2 starts at address 0010:0000h=00100h and ends at 0010:FFFFh=100FFh.

➢ **Observations:**

- The segments start at every 10h = 16 bytes

- The starting address of a segment always ends with 0h.

- We call 16 bytes a paragraph.

- An address divisible by 16 (ends with hex 0) is called a **paragraph boundary.**

Example-1

- **Physical address of a memory location is 1256Ah, find the address in *segment : offset* form for segment 1256h?**

  Physical address = Segment X 10h + Offset

  Offset = Physical address - Segment

Lets consider, X = offset in 1256h. Thus,

- 1256Ah=12560h [segment 1256 multiplied by 10] + X
- X = 1256Ah-12560h
- X = Ah
- X = 000Ah

Segment : offset = 1256:000Ah

Example-2

- **Physical address of a memory location is 1256Ah, find the address in segment : offset form for segment 1240h?**

Lets consider, X = offset in 1240h. Thus,

- 1256Ah=12400h [segment 1256 multiplied by 10] + X
- X = 1256Ah-12400h
- X = 16Ah
- X = 016Ah

Segment : offset = 1240:016Ah

Calculate the Segment Number

- **A physical address 80FD2h and offset BFD2h is given. Calculate the segment .**

  Physical address = Segment X 10h + offset

Thus,

- Segment X 10h = Physical address - offset
- Segment = (Physical address - offset) / 10h
- Segment = (80FD2h - BFD2h)/10h
- Segment = (75000h)/10h

  Segment = 7500h

Task

- Find the physical address of memory location **0A51:CD90h**.

- A memory location has physical address **4A37Bh**. Compute the offset if segment number is **40FFh**.

- Compute the Segment if offset number is **123Bh**.

Program Segments

- Machine language program consists instruction and data.
- Processor uses stack to implement procedure calls.
- The program code are loaded into **Code Segment (CS)** of memory.
- Data are loaded into **Data Segment (DS)** of memory.
- Stack are loaded into **Stack Segment (SS)** of memory.
- The 8086 uses four segment registers (CS,DS,SS, ES) to **hold segment numbers**.
- Any program needs access for second data segment may use **Extra Segment (ES)**.

Program Segments

- A memory segment does not necessarily occupy the entire 64KB in a memory segment.
- Programs segment of less than 64KB are placed close together due to its overlapping nature.
- At any given time, only four memories are active and thus these four segments are accessible.
- However, **contents** of memory segments can be modified by a program to **address different segments.**

Pointer and Index Registers

- The registers **SP, BP, SI, and DI** usually **point** to the memory locations.
- Registers **contain the offset address** of Memory location.
- Unlike segment registers, **Index** registers can be used in **arithmetic** and other operations.

Stack Pointer (SP)

- The stack pointer (**SP**) is used **together with SS** to **access** the stack segment.

Base Pointer (BP)

- BP is used to access data on the stack.
- Unlike **SP, BP** can be used to access data in the other segments.

Source Index (SI)

- **SI** is used to **point to memory locations** in the data segment addressed by **DS**.

- Consecutive memory locations can be accessed by **incrementing** the content of SI.

Destination Index (DI)

- **DI** is also used to point memory location.

- **String operations** use **DI** to access memory locations addressed by ES.

Memory Recall

- **What is the primary difference between Index registers and segment registers?**

Instruction Pointer (IP)

- The memory registers are for **data access**.
- The 8086 uses **CS** and **IP** registers to **access instructions.**
- **CS** contains the **segment number** and IP contains the **offset** of next register.
- IP is updated each time after an instruction execution to **point to the next pointer**.
- Unlike other registers, IP can **not be directly** manipulated by an instruction. (i.e. an instruction may **not** contain IP as its **operand**.)

FLAGS Register

- **FLAGS** register is used to **indicate the status** of the microprocessor.
- Indication is done by setting of **individual bits** [flags].
- There are two kinds of FLAGS

  ➢ **Status flags: Reflect the result** of an instruction executed by the processor. [More: chapter-5]

  e.g. If AX-BX results to **0**, the **ZF** (Zero Flag) is set to **1** (True)**.**

  ➢ **Control flags: Enable** or **Disable** certain operations of the processor

  e.g. if **IF (Interrupt Flag)** is cleared (set to 0), inputs from keyboard are ignored by the processor. [More: chapter-11]

The Organization of the PC

- A computer is made up of both Hardware and Software.
- Software controls the hardware operations.
- Thus, we need to understand the coordination of hardware and software to completely understand the operation of computer.

The Operating System

- The purpose of **OS** is to **coordinate** the operations between all the devices of the computer system.

- **The OS functions:**

  - Reading and executing the command typed by users.
  - Performing I/O operations.
  - Generating error messages.
  - Managing memory and other resources.

  - DOS was designed for 8086 processors.
  - DOS could manage only 1 MB memory and doesn’t support multitasking.

DOS OS

- DOS performs reading and writing information on disk.
- Programs and other information stored on a disk are organized into **files.**
- Each file has a **file name** [within 1 t0 8 characters.] followed by an **extension**.
- The extension is used for file type.

BIOS

- The BIOS routine perform I/O operation for the PC.
- However, DOS routine operates over the entire PC family.
- BIOS routines are **machine specific.**
- BIOS also performs **circuit checking** and **loading** of DOS routine.
- The address of the BIOS routines is called **interrupt vectors.**

Memory Organization of the PC

- 8086/8088 processor is capable of addressing 1MB or memory.

- But, all the memory **can not be used** by application program as some memory locations have special meaning to the processor.

    e.g. First KB (00000h to 003FFh) is used for interrupt vectors.

- Other memory locations are reserved by IBM for special purposes (i.e. BIOS routines and Video display memory).

Memory Partition into Disjoint Segments

- Partitioning the memory into disjoint segments. [Ref: Figure: 3.4]
- start **Segment 00000h= 0000:0000** ends at **Segment 0FFFFh**
- start **Segment 10000h=1000:0000** ends at **Segment 1FFFFh**
- start **Segment 20000h=2000:0000** ends at **Segment 2FFFFh**

… … …

- start **Segment F0000h** ends at **Segment FFFFFh** [Total 16 disjoint segments]
- The Only first 10 disjoint memory segments are used by DOS for **loading** and **running** application programs.
- The ten segments **0000h to 9000h** gives 640KB.
- The memory sizes of 8086/8088 based PCs are given in terms of these memory segments.

e.g. a PC with 512-KB memory has **eight** of these memory segments.

Memory Map of the PC

BIOS	F0000h
Reserved	E0000h
Reserved	D0000h
Reserved	C0000h
Video	B0000h
Video	A0000h
Application Program Area	
DOS	
BIOS and DOS data	00400h
Interrupt Vectors	00000h

Start-up Operation

- When PC is powered up, the 8086/8088 processor is put in reset state.
- The register CS = FFFFh and IP = 0000h.
- First instruction it executes, is located at FFFF0h.
- This memory location is ROM.
- ROM contains an instruction that transfers control to the starting point of the BIOS routines.
- The BIOS routine first check for system and memory errors.
- BIOS routines then initialize the interrupt vectors and BIOS data area.
- Finally, BIOS loads the operating system from the system disk.
  - step-1: BIOS loads boot program
  - step-2: Boot program loads the actual operating systems routines
- Boot program is named so because computer pulling itself by bootstraps.
- Once OS is loaded into memory, COMMAND.COM is given control

References

- Assembly Language Programming and Organization of the IBM PC, Ytha Yu and Charles Marut, McGraw Hill, 1992. (ISBN: 0-07-072692-2).
- https://www.britannica.com/technology/DNA-computing
- https://www.includehelp.com/embedded-system/memory-organization-in-the-8086-microprocessor.aspx

Books

- Assembly Language Programming and Organization of the IBM PC, Ytha Yu and Charles Marut, McGraw Hill, 1992. (ISBN: 0-07-072692-2).
- Essentials of Computer Organization and Architecture, (Third Edition), Linda Null and Julia Lobur
- W. Stallings, “Computer Organization and Architecture: Designing for performance”, 67h Edition, Prentice Hall of India, 2003, ISBN 81 – 203 – 2962 – 7
- Computer Organization and Architecture by John P. Haynes.