CET 302

Lecture #2 (8/31/04)

Chapter #2 in text

2.1 Microarchitecture of the 8088/8086 Microprocessor

Microarchitecture – internal to the processor

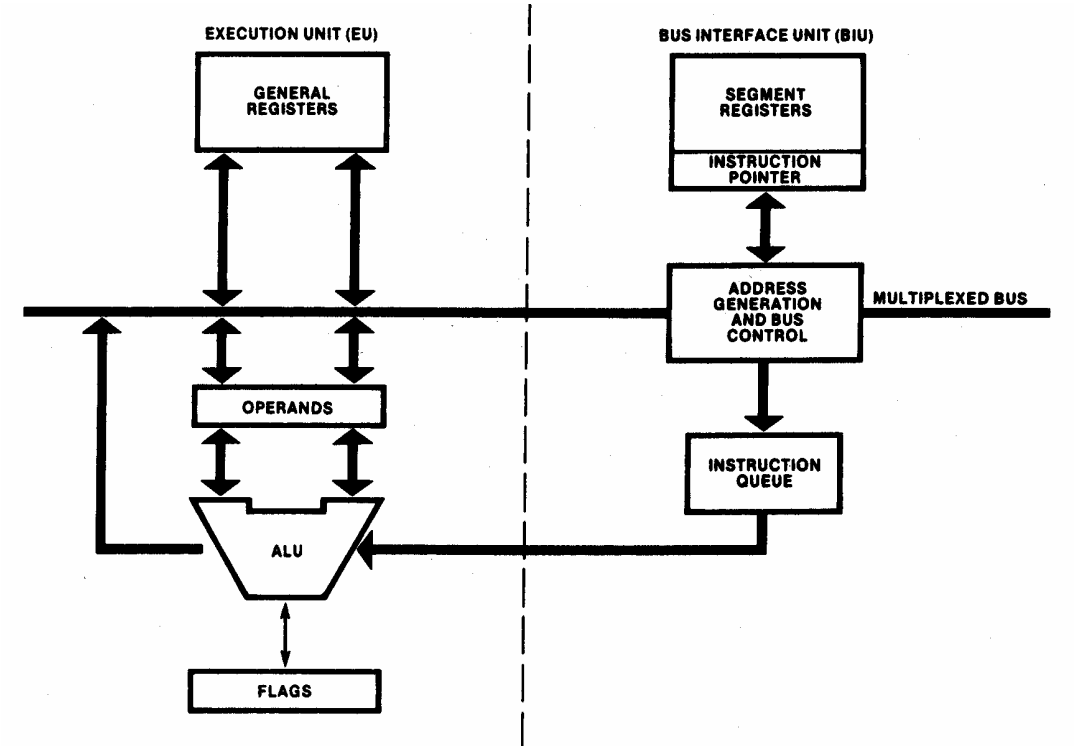


Figure 2-1

Prior to 8086 processors, the processors microarchitecture only performed one function at a time.

To improve speed/performance, the 8086/8088 processors started to do "Parallel Processing" (***explain quotes***)

In Fig 2-1, there are two processing units. The Execution unit (EU) and the Bus Interface Unit (BIU)

Bus interface Unit (BIU)

The main responsibility of the BIU is to store and retrieve data as the commanded by the EU, but it also maintains a four byte queue that is used to store the next instructions. The BIU maintains the queue by monitors the CS and IP registers to find out where the next instruction is stored. Intel designed the BIU in this manner, because they wanted to have the next instruction waiting in the queue as soon as the EU completes the previous instruction. This made the 8088 much faster than their predecessors, because the older processors did not use a queue for the next instructions.

http://www.cs.uregina.ca/~haughian/cs250/chpt4.htm

bus

Last modified: Wednesday, February 05, 2003

(1) A collection of wires through which data is transmitted from one part of a computer to another. You can think of a bus as a highway on which data travels within a computer. When used in reference to personal computers, the term *bus* usually refers to *internal bus*. This is a bus that connects all the internal computer components to the CPU and main memory. There's also an expansion bus that enables expansion boards to access the CPU and memory.

All buses consist of two parts -- an address bus and a data bus. The data bus transfers actual data whereas the address bus transfers information about where the data should go.

http://www.pcwebopedia.com/TERM/b/bus.html

The Execution unit (EU)

The execution unit controls everything that goes on in the processor. The EU retrieves the machine language instruction from the Instruction queue, it then deciphers the instruction and see that the correction action is taken. If an instruction requires external data, such as a memory location, the EU request the data from the Bus interface unit, or the instruction may need to use data that is being stored in one of the registers, in this case the EU makes sure that the correct register is being used.

If the instruction requests the use of the Arithmetic and Logic unit, the EU passes the data to the Arithmetic and Logic unit, and instruction on what operation to perform on the data. The Arithmetic and Logic unit then passes the result of the operation back to the EU, where it stores it, either in a memory location or a register. The EU does not do any arithmetic or logical operations, this left up to the Arithmetic and Logic unit.

http://www.cs.uregina.ca/~haughian/cs250/chpt4.htm

Arithmetic and Logic unit (ALU)

The ALU contains circuitry that can add two numbers together, word or byte form. It also contains circuitry that can add or subtract one to a byte or word, and shifting of bits or rotate bits, in a word or a byte.

The ALU is also capable of doing logical operations, such as; OR, AND, NOT, and XOR. Subtraction is done by circuitry that uses two's complementary functions.

The ALU is also responsible for doing bit operations, such as, shifting right or left, or rotating a byte or word. The ALU performs multiplication with a combination of additions and shifts, division is done in a similar manner.

When the ALU finishes an operation it also responsible for setting specific bits in the flag register, these bits are set according to the result of the operation.

http://www.cs.uregina.ca/~haughian/cs250/chpt4.htm

2.2 Software Model of the 8088/8086 Microprocessor

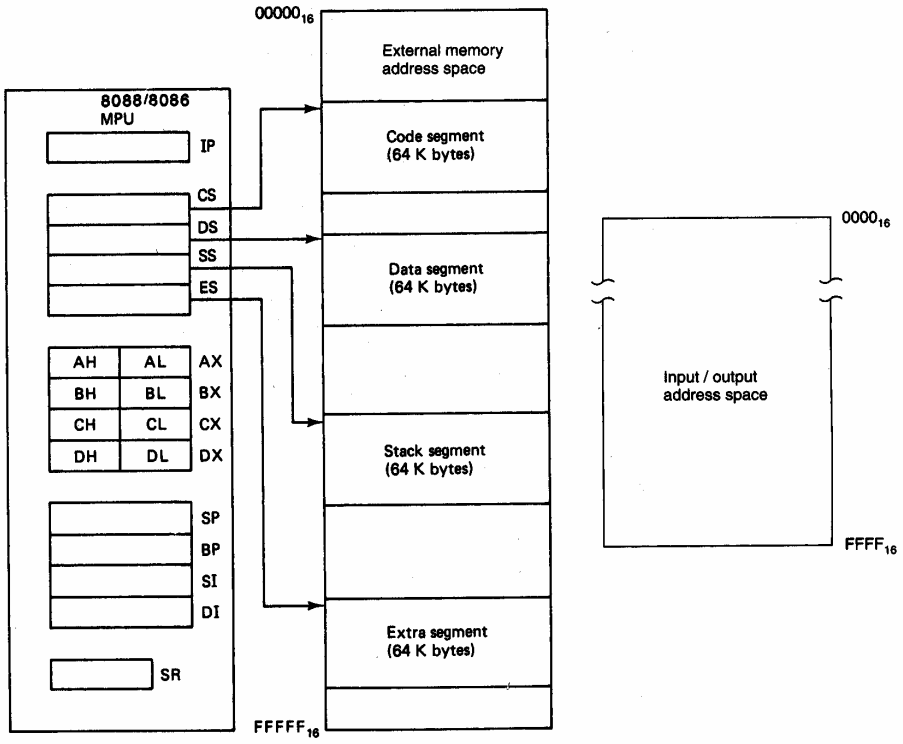
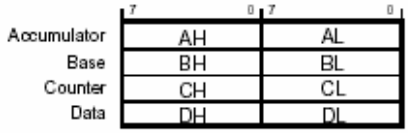


Figure 2-2

(Sections 2.7-2.10)

AX, BX, CX, DX

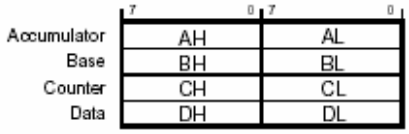
General Purpose Registers



- Can Be Used Separately as 1-byte Registers
  AX ← AH:AL
- Temporary Storage to Avoid Memory Access
  - Faster Execution
  - Avoids Memory Access
- Some Special uses for Certain Instructions

AX, BX, CX, DX

General Purpose Registers - Some Specialized Uses



- AX, Accumulator
  - Main Register for Performing Arithmetic
  - mult/div must use AH, AL
  - "accumulator" Means Register with Simple ALU
- BX, Base
  - Point to Translation Table in Memory
  - Holds Memory Offsets; Function Calls
- CX, Counter
  - Index Counter for Loop Control
- DX, Data
  - After Integer Division Execution - Holds Remainder

CS, DS, ES, SS - Segment Registers

Contains "*Base Value*" for Memory Address

- CS, Code Segment
  - Used to "point" to Instructions
  - Determines a Memory Address (along with IP)
  - Segmented Address written as CS:IP
- DS, Data Segment
  - Used to "point" to Data
  - Determines Memory Address (along with other registers)
  - ES, Extra Segment allows 2 Data Address Registers
- SS, Stack Segment
  - Used to "point" to Data in Stack Structure (LIFO)
  - Used with SP or BP
  - SS:SP or SP:BP are valid Segment Addresses

IP, SP, BP, SI, DI - Offset Registers

Contains "*Index Value*" for Memory Address

- IP, Instruction Pointer
  - Used to "point" to Instructions
  - Determines a Memory Address (along with CS)
  - Segmented Address written as CS:IP
- SI, Source Index; DI, Destination Index
  - Used to "point" to Data
  - Determines Memory Address (along with other registers)
  - DS, ES commonly used
- SP, Stack Pointer; BP, Base Pointer
  - Used to "point" to Data in Stack Structure (LIFO)
  - Used with SP or BP
  - SS:SP or SP:BP are valid Segment Addresses

These can also be used as General Registers !!!!!!

8086/8088 Register File (cont)

Flags Register

15															0
x	x	x	x	OF	DF	IF	TF	SF	ZF	x	AF	x	PF	x	CF

Status and Control Bits Maintained in Flags Register

- Generally Set and Tested Individually
- 9 1-bit flags in 8086; 7 are unused

http://www.ece.msstate.edu/~reese/EE3724/lectures/x86pmodel/x86pmodel.pdf

Status Registers:

Status Flags

1. *The carry flag (CF):* CF is set if there is a carry-out or a borrow-in for the most significant bit of the result during the execution of an instruction. Otherwise, CF is reset.
2. *The parity flag (PF):* PF is set if the result produced by the instruction has even parity—that is, if it contains an even number of bits at the 1 logic level. If parity is odd, PF is reset.
3. *The auxiliary carry flag (AF):* AF is set if there is a carry-out from the low nibble into the high nibble or a borrow-in from the high nibble into the low nibble of the lower byte in a 16-bit word. Otherwise, AF is reset.
4. *The zero flag (ZF):* ZF is set if the result produced by an instruction is zero. Otherwise, ZF is reset.
5. *The sign flag (SF):* The MSB of the result is copied into SF. Thus, SF is set if the result is a negative number or reset if it is positive.
6. *The overflow flag (OF):* When OF is set, it indicates that the signed result is out of range. If the result is not out of range, OF remains reset.

Control Flags

1. *The trap flag (TF):* If TF is set, the 8088 goes into the *single-step mode* of operation. When in the single-step mode, it executes an instruction and then jumps to a special service routine that may determine the effect of executing the instruction. This type of operation is very useful for debugging programs.
2. *The interrupt flag (IF):* For the 8088 to recognize *maskable interrupt requests* at its interrupt (INT) input, the IF flag must be set. When IF is reset, requests at INT are ignored and the maskable interrupt interface is disabled.
3. *The direction flag (DF):* The logic level of DF determines the direction in which string operations will occur. When set, the string instruction automatically decrements the address; therefore, the string data transfers proceed from high address to low address. On the other hand, resetting DF causes the string address to be incremented—that is, data transfers proceed from low address to high address.

The 8088 and 8086 Microprocessors 4th ed page 46

Memory and IO Space (sec 2.3 [part] and 2.13)

The software model shows a memory space (addresses 00000h to FFFFFh) and IO Space (0000h to ffffh)

Memory Map / IO Map (not that in the INTEL WORLD they are two separate maps [not true in all processors)

Explain the differences between memory and IO.

Dedicated, Reserved and General-Use Memory (sec 2.6)

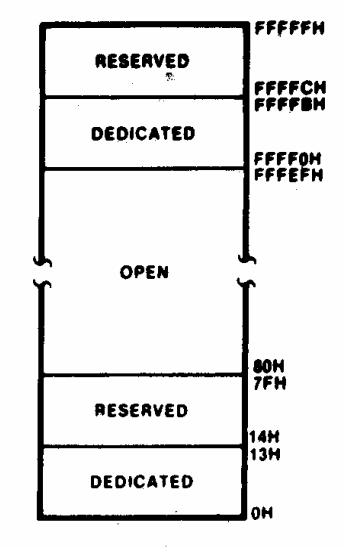


Fig 2-14

Addresses	Description
00000-00013H	Dedicated interrupts
00014-0007FH	Reserved
00080-FFFEFH	General use memory
FFFF0-FFFFBH	Dedicated functions
FFFFC-FFFFFH	Future expansion (do not use)

The most important dedicated function is the **HARDWARE RESET JUMP INSTRUCTION** found at memory location FFFF0H

Addressing (Sec 2.11)

A segment and offset describe a **logical address** But the 8088 processors uses physical addresses that are 20 bits long. How does the processor create a 20 bit address when it only has 16 bit registers (IP, Offset Registers, Seg Registers)?

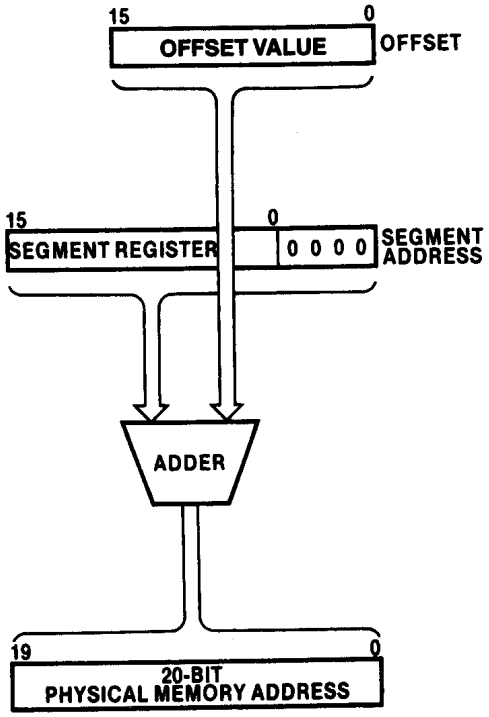


Fig 2-18

Storing Values (sec 2.3 [part] 2.4)
~~Aligned vs Misaligned~~

Storing 16 bit numbers:

Address	Mem (bin)		Mem (hex)
0072C	11111101		FD
0072B	10101010		AA

16 bit value would be 1111110110101010 or FDAAH and the address would be 0072BH
So if you load a 16 bit register from memory location 072BH it would contain FDAAH

Data types

Unsigned Integer 8 bit value (0 - 255 dec or 0-FFh)
Unsigned Integer 16 bit (word) value (0 - 65535 or 0-FFFFh)
Unsigned Double Word value (0 - 4,294,967,295 dec or 0 - FFFFFFFFh)

Signed integer 8 bit value (+127 to – 128dec) (Stored in two's complement)
Signed integer 16 bit value (+32767 to -32768 dec)
Signed Double word value (+2,147,483,647 to – 2147483648 dec)

Binary Coded Decimal (BCD)

The 8 bits are broken into 2 nibbles (4 bits)
Each nibble then represents 1 decimal digit (0 – 9)

ASCII

Way of storing characters into memory (see ascii table)

Many other encoding schemes for data

(we will come back to section 2.12 in a latter lecture (The Stack))

Chapter Two
HW

Pages 54-57

Questions: 7, 11, 14 (w/o aligned / misaligned), 15 (w/o aligned / misaligned), 30-33, 37, 55, 56

Additional Question: What determines the value of each of the status flags, when do these values change?