Organization of Intel Microcomputers

Module 3

CS 272

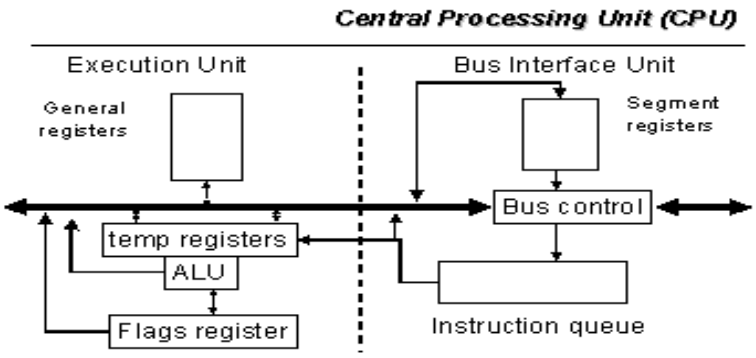
Sam Houston State University

Dr. Tim McGuire

Intel 8086 Family of Microprocessors

- All of the Intel chips since the 8086 have similar architectures and use the same core assembly language
- Learning the assembly language for the 8086 provides the basis for understanding the assembly language for the entire family of Intel chips.
- Microprocessor Features

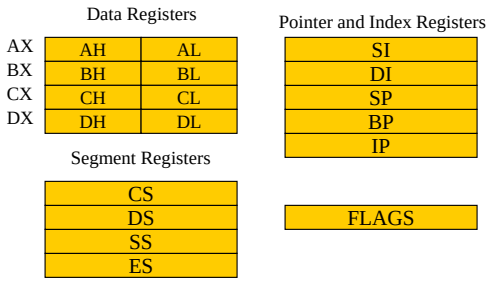
Organization of the 8086 Microprocessor



8086 Registers

- Information inside the microprocessor is stored in high-speed temporary memory locations called registers (fourteen 16-bit registers)
- *data registers* hold data for an operation
- *address registers* hold the address of an instruction or data
  - The address registers are divided into *segment*, *pointer*, and *index* registers
- a *status register* (called FLAGS) keeps the current status of the processor

8086 Registers



General Purpose, or Data Registers

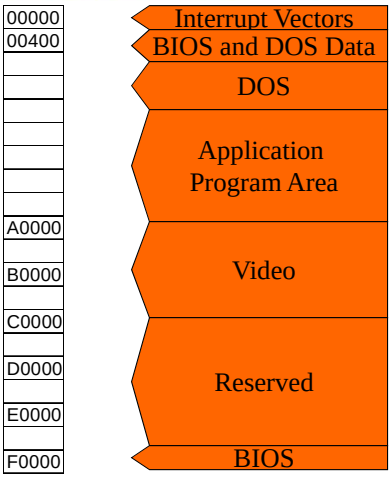
- The data registers may be used for general purposes, however each has special uses
  - AX : Accumulator
    - arithmetic, logic, data transfer
  - BX : Base Register
    - addresses
  - CX : Count Register
    - loop counter
  - DX : Data Register
    - multiplication/division
- Each byte of the 4 data registers can be accessed independently
  - AH, AL, BH, etc., refer to the high and low bytes of the registers
  - These are referred to as 8-bit registers, but remember they are part of an existing register

Pointer and Index Registers: SP, BP, SI, DI

- SP (*stack pointer*) points to the top of the processor's stack
- BP (*base pointer*) usually accesses data on the stack
- SI (*source index*) used to point to memory locations in the data segment
- DI (*destination index*) performs same functions as SI.
- DI and SI are often used for string operations

Memory

- 8086 - 1 megabyte of memory (2^{20} bytes)
- Each byte is accessed by specifying an address (00000h through FFFFFh)
- 20-bit addresses must be formed from 16-bits of information



Memory Segmentation

- Memory segments are a direct consequence of using a 20 bit address in a 16 bit processor
- Memory is partitioned into 64K (2^{16}) segments
- Each segment is identified by a 16-bit segment number ranging from **`0000h-FFFFh`**
- Within a segment, a memory location is specified by a 16-bit ***offset*** (the number of bytes from the beginning of the segment)
- The Segment:Offset address is a *logical address*

Segment:Offset Addresses

- **`A4FB:4872h`** means offset **`4872h`** within segment **`A4FBh`**
- To get the physical address, the segment number is multiplied by 16 (shifted 4 bits to the left) and the offset is added
- **`A4FB0h + 4872h = A9822h`** (20 bit physical address)
- There is a lot of overlap between segments; a new segment begins every 16 bytes (addresses ending in 0h)
- We call these 16 bytes a ***paragraph***
- Because segments may overlap, the segment:offset address is not unique

Segment Register Example

- 20-bit addresses are formed by pairing a segment register with another register in a ***segment:offset address***
  - The segment register holds the segment number
  - The other register holds the offset
  - Address = (segment number)*16+offset
- Example CS: 010C IP: 14D2
  - Address = 010Ch*10h+14D2h = 010C0h+14D2h
    Address = 02592h
- Since segments are 64K they can start at any ***paragraph boundary***

Segment Registers: CS, DS, SS, ES

- CS (*code segment*) addresses the start of the program's machine code in memory
- DS (*data segment*) addresses the start of the program's data in memory
- SS (*stack segment*) addresses the start of the program's stack space in memory
- ES (*extra segment*) addresses an additional data segment, if necessary

Program Segments

- During program execution, the segment registers are only changed if memory not currently accessible in an active segment must be accessed
- Program bytes are arranged into distinct segments for convenience
  - CS -> segment containing machine instructions
  - SS -> segment containing storage for the stack
  - DS -> segment containing data values and storage
  - ES -> segment for additional data or special memory operations
- Programmers must be aware of this organization

Instruction and Stack Pointers

- IP contains the address of the next instruction to be executed
  - IP specifies an offset into the CS segment
  - IP is not the operand of any instruction
- SP points to the top item on the stack
  - SP is an offset into the SS segment
  - SP can be used as an operand in some instructions

Instruction Pointer: IP

- The 8086 uses registers CS and IP to access instructions
- The CS register contains the segment number of the next instruction and the IP contains the offset
- The IP is updated each time an instruction is executed so it will point to the next instruction
- The IP is not directly accessible to the user

BP and Index Registers

- BP is a Base Pointer
  - Specifies an offset into any segment, but most commonly the Stack segment
- SI and DI are called Index registers
  - Normally specify an offset into the Data segment, but they can be used as offsets into any segment
  - Sometimes hold a number to be added to the address of an array (index)

Flags

- Individual bits are used to store the status of the microprocessor
  - Bits are set or cleared as the result of many operations
  - Bits may be affected indirectly (by the execution of an instruction) or directly by an instruction designed to access the status word

The FLAGS register

- Indicates the status of the microprocessor
- Two kinds of flag bits: ***status flags*** and ***control flags***
- Status flags reflect the result of an instruction, e.g., when the result of an arithmetic operation is 0, ZF (***zero flag***) is set to 1 (***true***)
- Control flags enable or disable certain operations of the processor, e.g., if the IF (***interrupt flag***) is cleared (set to 0), inputs from the keyboard are ignored by the processor

Operating System and Hardware Features

- MS-DOS
- COMMAND.COM
- BIOS
  - Interrupt Vectors
  - Video Memory
- IO Ports

MS-DOS

- The operating system coordinates the operation of the computer systems
- Its functions include:
  - reading and executing the commands typed by the user
  - performing I/O operations
  - generating error messages
  - managing memory and other resources

COMMAND.COM

- The DOS routine that services user commands
- Responsible for generating the DOS prompt and reading user commands
- Determines whether the command typed is an ***internal*** or an ***external*** command
- Internal commands are in COMMAND.COM; External commands are applications which may or may not be part of DOS

BIOS

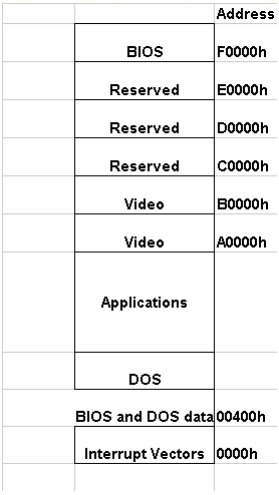
- **B**asic **I**nput/**O**utput **S**ystem
- System routines stored in ROM which are not destroyed when the power is off
- BIOS routines may vary slightly from one machine model to the next

Interrupt Vectors

- To let DOS and other programs use the BIOS routines, the addresses of the BIOS routines, called ***interrupt vectors***, are placed in memory, starting at 00000h

Video Memory

- Segments A000h and B000h are used for video display memory
- Segment F000h is a special circuit because it references ROM instead of RAM -- it holds the actual BIOS routines



IO Ports

- Special addresses are designated for I/O devices, such as keyboard, monitor, printers, modems, etc.

System Startup

- Reset state
  - CS = FFFFh
  - IP = 0000h
- Executes an instruction in ROM that transfers to a BIOS routine
- System and memory check
- Initialize interrupt vectors and BIOS data
- Load operating system from disk
  - boot program from boot sector
  - DOS kernel
- Load and execute command.com