CS430 Computer Architecture

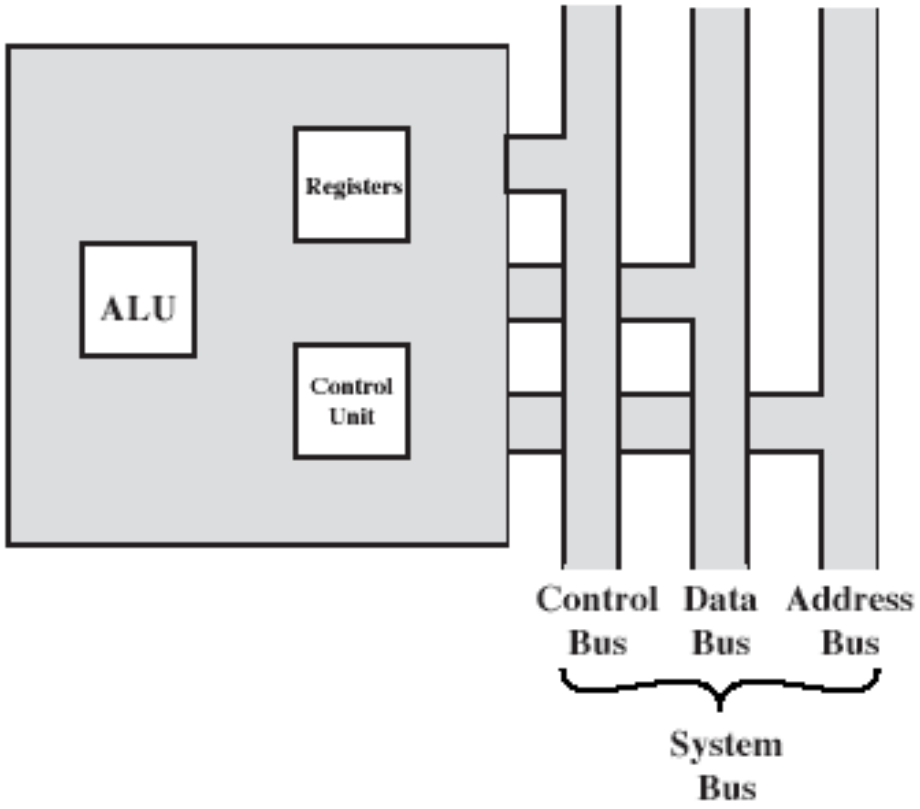
Spring 2015

Chapter 14
Processor Structure and Function

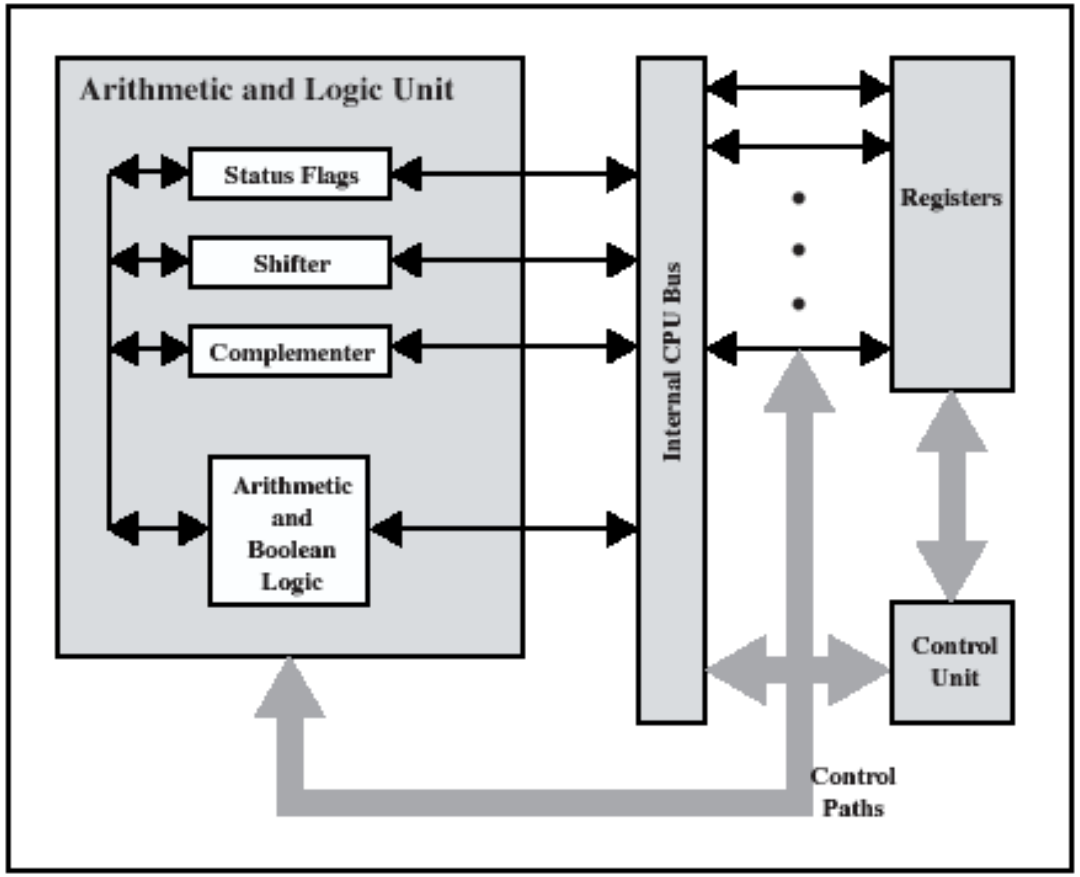
- Reading: pp. 483-503

- Processor Requirements
  - Fetch Instruction
  - Decode Instruction
  - Fetch Data
  - Process Data
  - Write Data

Simplified Processor View



Internal CPU Structure



Registers

- user-visible – available to the programmer to minimize memory references
  1. General Purpose – can be used for a variety of purposes
  2. Data – used to hold data but cannot be used in the calculation of an operand address
  3. Address – can be somewhat general purpose or used for a particular addressing mode (e.g. segment registers, index registers, stack pointer)
  4. Condition Codes – bits set by the processor as the result of a particular operation

- Some processors use condition codes and some do not.

Condition Codes

Advantages	Disadvantages
1. Because condition codes are set by normal arithmetic and data movement instructions, they should reduce the number of COMPARE and TEST instructions needed. 2. Conditional instructions, such as BRANCH are simplified relative to composite instructions, such as TEST AND BRANCH. 3. Condition codes facilitate multiway branches. For example, a TEST instruction can be followed by two branches, one on less than or equal to zero and one on greater than zero.	1. Condition codes add complexity, both to the hardware and software. Condition code bits are often modified in different ways by different instructions, making life more difficult for both the microprogrammer and compiler writer. 2. Condition codes are irregular; they are typically not part of the main data path, so they require extra hardware connections. 3. Often condition code machines must add special non-condition-code instructions for special situations anyway, such as bit checking, loop control, and atomic semaphore operations. 4. In a pipelined implementation, condition codes require special synchronization to avoid conflicts.

Control & Status Registers

- Used by the processor's control unit and by privileged OS programs to control program execution.

- We've already discussed uses for the:
  - Program Counter (PC)
  - Instruction Register (IR)
  - Memory Address Register (MAR)
  - Memory Buffer Register (MBR)

x86 Program

Consider the following x86 assembly language program:

```
13CF:0100 B80000        MOV     AX,0000
13CF:0103 BB0000        MOV     BX,0000
13CF:0106 40            INC     AX
13CF:0107 01C3          ADD     BX,AX
13CF:0109 3D0A00        CMP     AX,000A
13CF:010C 75F8          JNZ     0106
13CF:010E 90            NOP
```

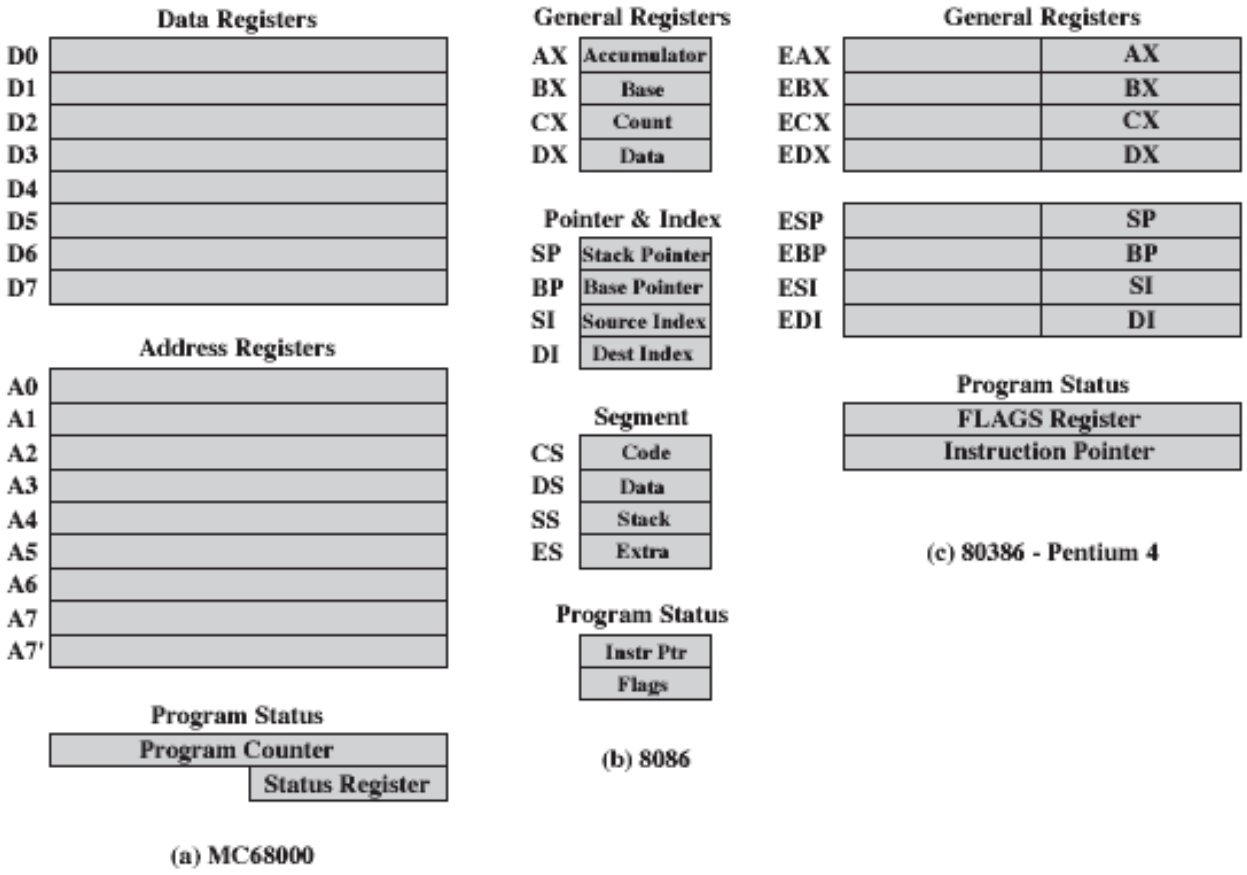
x86 Questions

1. What does the program do?
2. What is the initial PC value?
3. What is the PC in the x86 world?
4. What is the PC after executing the 1st statement?
5. Which of the above statements affect the condition codes?
6. Why is the machine language for JNZ 0101 equal to 75F8?

x86 Questions

7. How is the PC updated after execution of the statement JNZ 0106?
8. What is the relative address range for the entire program?
9. What is the physical address range for the entire program?

Register Organization



Problem

- Fill in the blanks below

```
01ff: 0123  90      top:    nop
01ff: 0124  41              inc cx
01ff: 0125  03 d1           add dx,cx
01ff: 0127  83 fa 14        cmp dx,20
01ff: 012a  74 __           je out1
01ff: 012c  41              inc cx
01ff: 012d  48              dec ax
01ff: 012e  75 __           jne top
01ff: 0130  90      out1:   nop
```