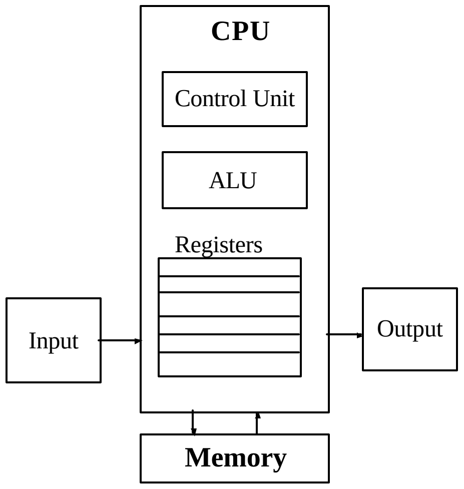
Lecture 5: Computer Architecture

- Computer Architecture
  - Registers
  - Flags
  - Address Calculation

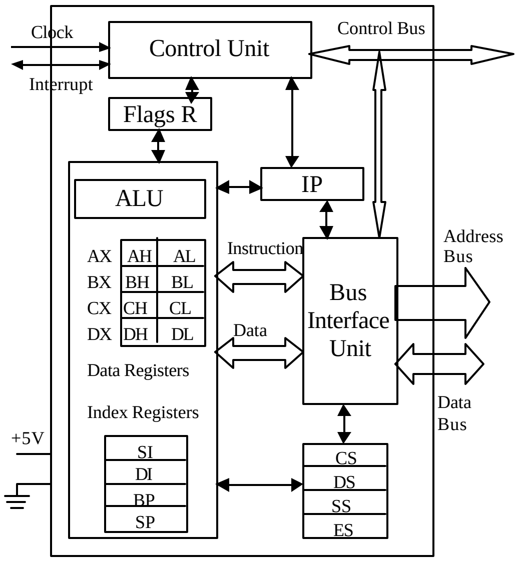
Computer Organization



Components

- Control Unit –
- Arithmetic logical unit (ALU) –
- Registers –
- Buses –

8086 Block Diagram

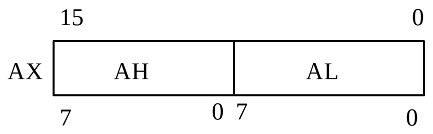


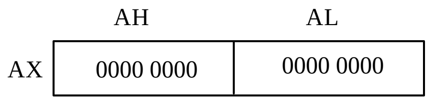
Registers

General Purpose Registers

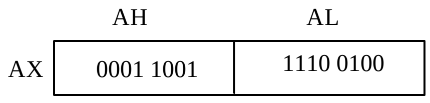
- Data registers, also known as general purpose registers: AX, BX, CX, DX
- Used for arithmetic operations and data movement
- Can be addressed as 16 bit or 8 bit values. For AX, upper 8 bits are AH, lower 8 bits are AL.
- Remember: when when a 16 bit register is modified, so are the corresponding 8 bit registers!

Example

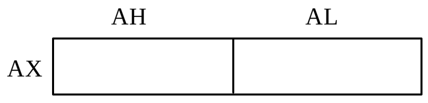




- move 0001 1001 1110 0100 to AX



- move 0011 1101 to AH



Special Attributes of GP Registers

- AX – accumulator
- BX – base
- CX – counter
- DX – data

Segment Registers

- Segment registers are used as base locations for program instructions, data, and the stack.
- All references to memory involve a segment register as the base location.

Segment Registers, cont.

- CS – code segment
- DS – data segment
- SS – stack segment
- ES – extra segment

Index Registers

- Index registers contain the *offsets* of data and instructions.
- Offset:
- Index registers are used when processing strings, arrays, and other data structures.

Index Registers, cont.

- BP – base pointer
- SP – stack pointer
- SI – source index
- DI - destination index

Status and Control Registers

- IP – instruction pointer
- Flags –

Status Flags

- Carry flag (CF)
- Overflow flag (OF)
- Sign flag (SF)

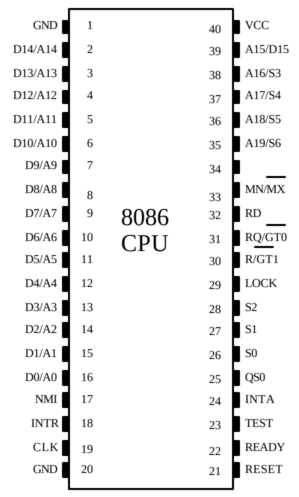
Status Flags, cont.

- Zero flag (ZF)
- Parity

Addressing

- Address: a number referring to an 8-bit memory location
- Logical addresses go from 0 to the highest location
- Logical addresses require translation into physical addresses
- For Intel (8086):

Pins on the 8086



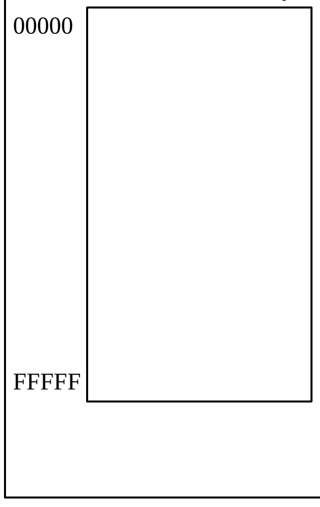
Addressing, cont.

- How do you have a 20-bit address with 16 bit registers?

Why Segment-Offset?

Data Segment

addressable memory on 8086



Example:
DS = 0100h
data seg. start?
data seg. end?

If BX contains the offset:

BX = 005Ah
EA = ?

Segment Register Combinations

- Code Segment – the CS register and IP (instruction pointer) are used to point to the next instruction.
- Stack – the SS register is used with the SP (stack pointer) or BP (base pointer)
- Data Segment – DS with BX, SI, or DI
- Extended Segment – BX, SI, or DI

More on Effective Addresses

- There's more than one way to get the same effective address!
- Example:
  - CS = 147Bh
  - IP = 131Ah
  - EA = 147B0 + 131A = 15ACAh

  or
  - CS = 15ACh
  - IP = 000Ah
  - EA = 15AC0 + 000A = 15ACAh

- If CS = 147B, what range of effective addresses can be referenced without changing the value in CS?

Homework 2

- Two parts:
  - Part 1: Use Debug to enter and run a simple machine code program
    - convert input data into 2's complement hex
    - enter data at the correct address
    - enter program at the correct address
    - run the program
  - Part 2: Write a simple machine code program, given pseudo-code
    - these instructions should be similar to those in the Part 1 problem.
    - enter and run the resulting program.

Part I - Example Program

Given below is a machine code program that calculates the sum of all the words in a given range of addresses in memory. The code expects that the lower bound of this range is specified in the BX register and the upper bound in the DX register. (BX holds the offset of the beginning of the data to be summed from the beginning of the data segment (DS). DX holds the offset of the last data element from the beginning of the data segment.) The sum gets stored in AX. The first 4 hex digits given on each line below represent the offset of the instruction from the beginning of the code segment. The digits after the dash are the machine code instructions. To the right are English explanations of the instructions.

```
0000 - 2BC0      subtract AX from itself (to make it 0)
0002 - 0307      add the word pointed to by BX to AX
0004 - 83C302    add 2 to BX (to point to the next word)
0007 - 3BD3      compare BX to DX
                 (compare sets internal flags that are used by
                 subsequent jump instructions)
 0009 - 7DF7     if DX >= BX, then jump back to the
                 instruction at 0002
000B - B8004C    this instruction and the next one return
                 control to DOS
000E - CD21
```

Part 1

- Convert input data into 2’s complement hex
  - use the techniques you used on HW2
- Enter data at the correct address
  - DS holds the segment
  - Look at how you did this in lab!
- Enter program at the correct address
  - CS holds the segment
  - IP holds the offset
  - Again – look at the lab!
- Run your program!