Administrivia

Syllabus

Course outline

0. What is Computer Engineering (CpE)?

- a. HARDWARE: Anything that hurts if you drop it on your foot
- b. SOFTWARE: Everything else.
- c. INTERFACES: Connections to the external world

- d. VAPORWARE: Something promised but not delivered (Windows 95)
- e. WETWARE: Living things

Computer engineering subsumes Hw/Sw interfaces.

1. What is Assembly Language?
2. Why Assembly language programming?

To talk to a computer (tell it what to do), you need to speak its language.

C, FORTRAN, PASCAL, LISP, PROLOG. High Level Languages (HLLs).
Prediction: In a few (~5) years English will do fine.

HLLs: Easy to use, HARDWARE INDEPENDANT

But Cpe is hardware DEPENDANT, after all Computer Engineers (Cpers) make it possible for HLLs to be independant of the hardware!

Hardware dependant means that each MICROPROCESSOR (mp), the brains of a computer, has its own language.

Block diagram of a computer

See figure 2.2 for block diagram of 8086 CPU

These mp languages are similar enough to each other that learning one language for a particular mp makes it real easy to learn the languages of other mps.

Mp languages are called MACHINE LANGUAGES. Because of the physics of the underlying hardware Machine languages are binary. 0's and 1's

Machine Languages are a real Pain to program in. To make things less painful; in the early days (prehistory) of computers (1940s) ASSEMBLY Language was invented. Assembly language is a mnemonic form of Machine langauge (see figure 3.1 in text)

We will be learning to program the INTEL 80XXX family of Mps.
8086, 8088, 80286, 80386, 80486, (Pentium is the latest)

Some other other companies that design Mps are
Motorola: M680xx family, PowerPC family: Apple, IBM
MIPS : RXX00 family, SGI, DEC
DEC : Alpha family
HP : PA-RISC family
SUN : Sparc family
...
To understand and use Assembly you need to understand how data/information and instructions are stored in a computer as 0's and 1's

Representing data

Positive Integers
Positive and Negative Integers
Characters
Floating point numbers

Positional Number Systems

1, 2, ..., 10,...45, 74312, 038

We use the DECIMAL number system

1 = 1 * 10^0

2 = 2 * 10^0

10 = 1 * 10^1   +   0 * 10^0

45 = 4 * 10^1   +   5 * 10^0

74312 =
7 * 10^4 + 4 * 10^3 + 3 * 10^2 + 1 * 10^1 + 2 * 10^0

038 = 0 * 10^2 + 3 * 10^1 + 8 * 10^1

In general a number

D_3 D_2 D_1 D_0 in decimal will be interpreted as
D_3 * 10^3 + D_2 * 10^2 + D_1 * 10^1 + D_0 * 10^0

Decimal systems are base 10; we use powers of 10

Digits used in decimal are 0,1,2,3,4,5,6,7,8,9

**If we use some other positional number system,
say with a base of 8, then we'd use powers of 8.**

Digits used in base 8 would be 0,1,2,3,4,5,6,7

In general in base B, we use the digits from
0 thru B - 1

and a number

$D_3\ D_2\ D_1\ D_0$ in base B will be interpreted as

$$D_3 * B^3 + D_2 * B^2 + D_1 * B^1 + D_0 * B^0$$

or more generally, a number

**$D_n\ D_{n-1}\ \ldots.\ D_3\ D_2\ D_1\ D_0$ in base B will be
interpreted as**

$$D_n * B^n + D_{n-1} * B^{n-1} + \ldots.. + D_3 * B^3 + D_2 * B^2 + D_1 * B^1 + D_0 * B^0$$

If we use base 2, the only digits are 0 and 1

1101 in base 2 is

1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 13 in decimal

If we use base 16, the only digits are 0 thru 15
???

how do we write 10, 11, 12, 13, 14, and 15 using only one position

10 -------> A
11 -------> B
12 -------> C
13 -------> D
14 -------> E
15 -------> F

2A in base 16 is

2 * 16^1 + A * 16^0 = 32 + A = 32 + 10 (decimal) = 42

Base 16 and Base 2 are very important when dealing with computers.

Base 2 is BINARY.
Base 16 is HEXADECIMAL or HEX
Base 8 (OCTAL) is also used, although its use is decreasing

Converting from one system to another

Decimal to Binary

Decimal to Hex

Decimal to Base B

Binary to Hex   and   Hex to Binary
!Easy!

Addition and Subtraction in Hex or Binary

We have been assuming that numbers can be of any length.
We have been assuming that there is space for the Carry.
What about Negative integers/numbers?

The solution is predicated on the fact that
computer memory is finite and organized in units

BITS
4 bits NIBBLES
8 bit BYTES
16 bit WORDS

Remember most Registers are 16 bits or 1 word sized

Bit numbering

For 1 byte
Bit 0 is rightmost bit
Bit 7 is leftmost bit

For 1 word
Bit 0 is rightmost bit
Bit 15 is leftmost bit

Suppose we are working with numbers that must fit in 8 bits.

Bit 0 corresponds to 2^0 Least significant bit (LSB)
Bit 1 corresponds to 2^1
Bit 2 corresponds to 2^2
...
Bit 7 corresponds to 2^7 Most significant bit (MSB)

With 16 bits Bit 0 is LSB
Bit 15 is MSB

so

Binary	Hex	decimal
01011101	5D	93
11111111	FF	255
00000000	00	0
00110011	33	51

But suppose we add

```
          11001001                  201
          01001111                   79
          ------------              ------
1         00011000                   24  but shd be 280
```

So addition (and subtraction) cause problems.

What about the sign? positive and negative integers??

**One idea is to use 1 bit out of the 8 to represent the sign
This is the SIGNED MAGNITUDE Representation**

We use the MSB to represent SIGN, the rest of the bits represent the MAGNITUDE

**0 means +ive
1 means -ive**

00000101   = +5

10000101   = -5

10000111   = -7

01111111   = +127

11111111   = -127

**00000000   = +0
10000000   = -0   ???**

Addition and subtraction:

Two rules:

**Rule 1: When adding two numbers of like sign
Add the two MAGNITUDES and assign the "like" SIGN
to the result**

Example: -50 + - 25 = -75

add two magnitudes = 75 ; sign is -ive so answer = -75

**Rule 2: When adding two numbers of unlike sign
Subtract the smaller magnitude from the larger magnitude
and assign the sign of the larger magnitude to the result**

Example: 25 + -50 = -25

subtract 50 - 25 = 25 assign sign of 50; answer = -25

**But there are problems with this kind of
representation and computers use a different kind
of representation call 2's complement.**

Next class......

Read Chapter 1 and Chapter 2 for thursday