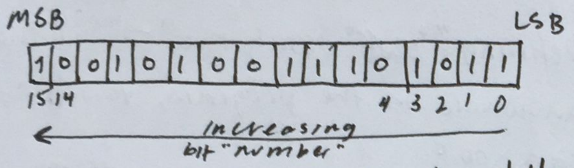
CS271 Week 1
-reading-

1.1-1.7

- assembler: source assembly code → machine code
- Linker: combines individual files produced by assembler into one executable
- MSB, LSB (Most/least significant bits) are as follows (16 bit num):



↳ MSB above would be the bit that would correspond with the highest change in value of the number represented ($2^{15} \cdot bit_{15}$)

- Units of Data size

unit	size (bytes)
kilobyte	2^{10} = 1024
megabyte	2^{20}
gigabyte	2^{30}
terabyte	2^{40} ~1.1 trillion bytes!
petabyte	2^{50}
exabyte	2^{60}
zettabyte	2^{70}
yottabyte	2^{80}

- Binary to Hex

If you have binary 00010110 split it up:

0001	0110	base 2
1	6	base 10
1	6	base 16

$\therefore 00010110_{2} = 16_{16}$

"x86" covers many 16, 32, 64 Bit processors. Started with Intel 8086, now on intel, Amd...

2.1-2.3

Instruction Execution Cycle

→ fetch instruction from Queue

→ CPU decodes instruction from binary

→ CPU Gathers operands from memory

→ CPU executes instruction

→ CPU stores result.

- Simplified: Fetch, Decode, Execute

Cache: stored instructions that are more quickly accessed than memory.

Memory fetch process

- CPU places address of mem on the address bus
- RD (read pin, CPU) changed
- Wait 1 clock cycle (mem spd ?= CPU spd)
- fetch data $\overset{DATA}{BUS}$ → Operand

Flags in the EFLAGS register

- Carry (CF): unsigned overflow
- Overflow (OF): signed overflow
- Sign (SF): result is negative
- Zero (ZF): result is zero
- Aux. Carry (AC): overflow from 0..3 to 0..4 or more (bits) of an 8 bit register.
- Parity Flag [PF]: LSB_{result} Even # of 1 bits

Floating Point

- used to require different processor
- Uses registers ST(0).. ST(7)

Memory types (2.4) Bonus!

- ROM: flash once, permanent thereafter
- EPROM: UV erasible (slow), reprogrammable
- DRAM: Main memory, must be refreshed every 1ms (except if its ECC)
- SRAM: High speed cache memory usually built into CPU's
- VRAM: Ram for a display, dual ported for reads and refresh at the same time
- CMOS RAM: where cmos (config) data is stored. kept alive by a battery.

3.1 - 3.5

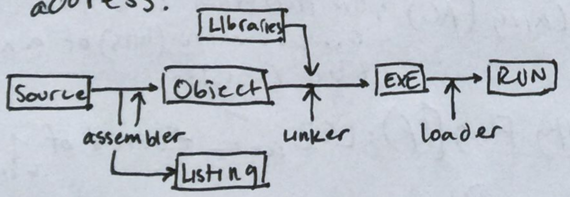
Assemble Link Execute Cycle

→ Programmer creates ASCII text file called program_source_code

→ Assembler makes an object file from program_source_code

→ Linker "Links" (copies) any linked or necessary libraries, and bundles them with the object file, creating an executable

→ Loader loads the program into memory, pointing the CPU (by setting a register) at the starting address.



DUP() operator

- allocates multiples of a variable's space in ~~sta~~ memory
- BYTE 20 DUP(0) = 20 bytes all Ø's
- BYTE 4 DUP ("STACK") = "STACK" x4 = 20 bytes
- BYTE 20 DUP (?) = 20 undefined bytes

x86 and Little Endianness

- x86 are Little Endian!!!
- Little Endian of 0x12345678 is:

offset	0000	0001	0002	0003
	78	56	34	12

smallest byte is END

"Defining" *stuff* using the "=" ‖ EQU = "=" :)

- anywhere in the program, you can do:

```
NAME = <value>10  ; maybe sticking with base 10 is best
```

- Later, this is valid, and neater

```
MOV  eax, NAME  ; Better than MOV eax, <value>
```

‖NOTE: $ is the current offset

List sizes

```
List_name   BYTE 10, 20, 30, 40
List_size = ($ - List) ; Cool, huh?
```

‖NOTE: VEWY VEWY BAD:

```
List_name   Byte 10, 20, 30, 40
var_17      BYTE 420
List_size = ($ - List); NOT list size!
```

- If using something larger than BYTE, Divide by the size multiple that type is above BYTE!

TEXTEQU

- name TEXTEQU <text>
- can be used quite funnily...