Appx C - Assembly language programming

When reading Appendix C, remember!

- Our gcc syntax is different than the text.
- This is an old Intel 8088 ISA, not our modern Intel architecture

I will highlight the important parts of Appx C in these notes.

C.1 Overview

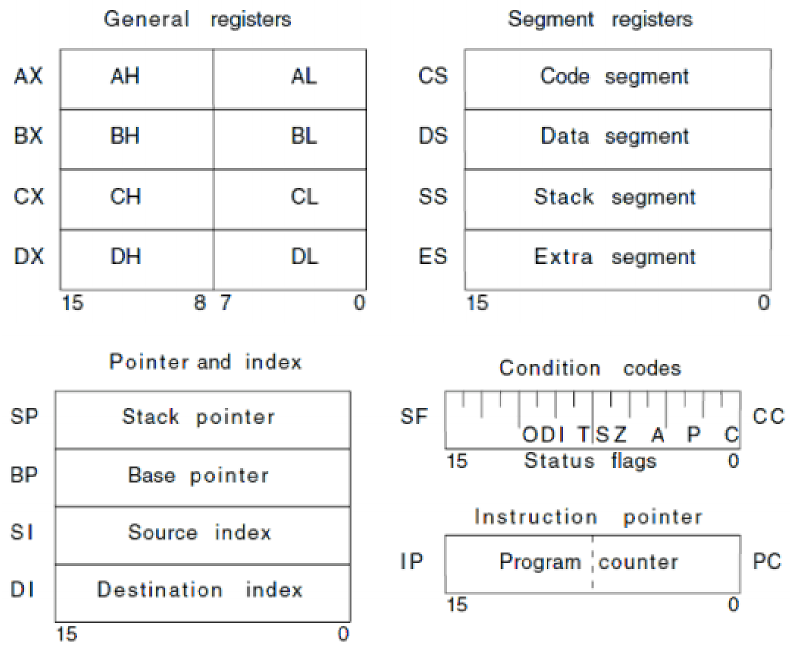
Definitions: mnemonics, labels, pseudo-instructions, assembler

C.2 The 8088 processor

Definitions: registers, program counter, code segment
Processor cycle is similar to our Ch 4 favorite, Mic-1:

1. Fetch assembly instr
2. PC++
3. Decode instr
4. Read data from memory or registers
5. Perform instr (datapath!)
6. Store results in memory or registers
7. Goto step 1

The 8088 registers:



C.3 Memory and addressing

4 memory segments:

- Code segment - your program
- Data segment - constants and global variables
- Stack segment - the stack for local variables and function parameters
- Extra segment - used as needed

The starting address of each segment resides in a register. Addresses are offsets from there.

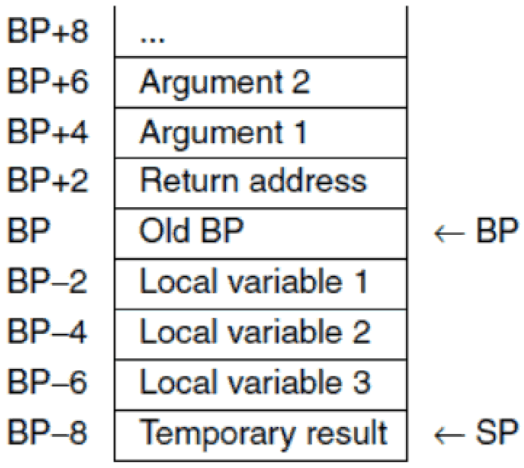
Addressing modes: register, data segment, stack segment

C.4 8088 Instruction set

Instruction types:

- Move, copy arithmetic ops - `mov, xchg, push, pop, add, sub, mul, div`
- Logical, bit, shift ops - `not, and, or, xor, shr, sal, rol, ror`
- Loop, string ops - `loop, movs, lods, stos, cmps`
- Jump, call instructions - `jump, jcc, call, ret`
- System calls - `sys`

What's the difference between a near jump and a far jump?
Frame pointer!!!



Quiz - What does push do? What does pop do? What is "Return address"? What is "Temporary result"? In our syntax... what is `-2(%ebp)`? `4(%ebp)`?