Reading 3

Brockman Textbook 2.5 Assembly Language and Machine Code

- Every microprocessor has a set of basic instructions called its instruction set that specifies the primitive operations that the microprocessor can perform.

- Each instruction in processor's intstruction set has a:
  - text representation called assembly language
  - binary representation called machine code.

C → assembly language → machine code

2.5.2 Assembly Language Program

- "gcc -S addsub.c" → compiles to an assembly language program
  - ↳ Statements starting with a "." are not instructions but directives that define symbols or give hints on how to assemble program
  - ↳ Names followed by ":" are labels for memory addresses where either instructions or data are stored
  - ↳ Names starting with "%" refer to limited, temporary storage locations inside the processor hardware called registers.
  - ↳ Register names inside of "( )" refer to memory addresses
  - ↳ "mov" instructions copy data between registers or between registers and memory.
  - ↳ ".LC0" is a label for where the memory location where the printf string is stored.

All for Intel x86 assembly language

2.5.3 Machine Code for Executable Program

- "objdump -d a.out" gives you info on the executable a.out
  - shows contents of executable in both machine code and assembly language

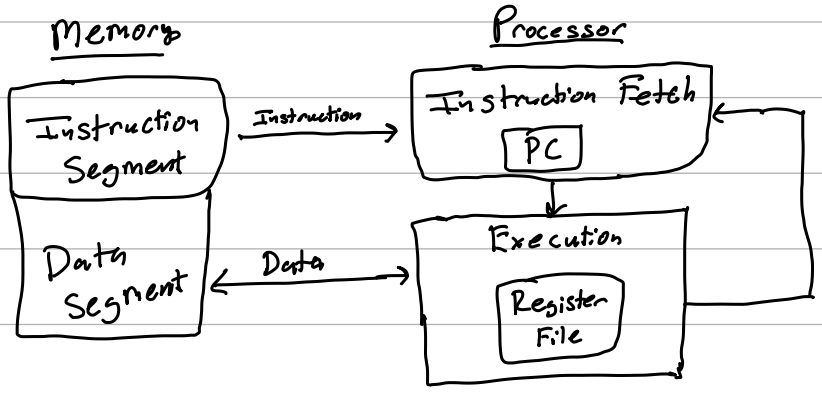
3.1 Basics of Computer Architecture and Organization

- A computer has 2 main hardware components:
  1. Memory: Stores instructions and data
  2. Processor: reads instructions from memory and then executes these to process the data

- A software tool called a compiler translates programs from a high-level language down to machine code.

- Interpreter: a kind of compiler that does the translation on the fly.

- Processor hardware organization has 2 main stages:
  1. Instruction fetch stage: fetches instructions from memory using a register called a program counter (PC) to keep track of which memory address to fetch next instruction from.
  2. Execution Stage: executes the instructions that process the data and memory, usually contains a register file that is much smaller and faster than memory to hold frequently used variables and intermediate results.



3.2.2 albaCore Assembler and Simulator

- The albaCore assembler albaasmh translates albaCore assembly language programs into a hexidecimal representation of machine code.

3.2.3.1 Operation and Instruction Encoding

- "Add register 1 and register 2, store result in register 5"
  - ↳ `add r5, r1, r2`
    - r5: destination (write)
    - r1: source a (read)
    - r2: source b (read)
- data words in albaCore are 16-bits wide.
- instructions in albaCore are 16-bits wide.
- 16 total instructions in albaCore instruction set.
  - ↳ sufficient to support much of C.

- Within a 16-bit instruction:
  - ↳ operation code (opcode): 4 most significant bits that identify the instruction.
  - ↳ load immediate instruction (ldi): loads an 8-bit constant into the 8 least significant bits of a 16-bit register and sets the upper bits of the registers to zeros.
    - ↳ requires several steps to load a 16-bit constant into a register.

- ".text" in albaCore indicates the following lines of codes are instructions.