CPU

- The *Central Processing Unit* (CPU) is the part of the computer system responsible for interpreting and executing programs
  - It contains a datapath (ALU, registers, interconnect) and a controller
  - Examples: Intel 486, Intel Pentium
- For now, we will look at only one CPU architecture: an *accumulator* machine
  - Instruction format:

7 5	4 0
opcode	address

  - Instructions: (*addr* = address)

    ADD *addr* — add value at *addr* to accum.

    SUB *addr* — subtr. value at *addr* from acc.

    MPY *addr* — mult. value at *addr* by acc.

    DIV *addr* — divide value at *addr* by acc.

    LOAD *addr* — load acc. with value at *addr*

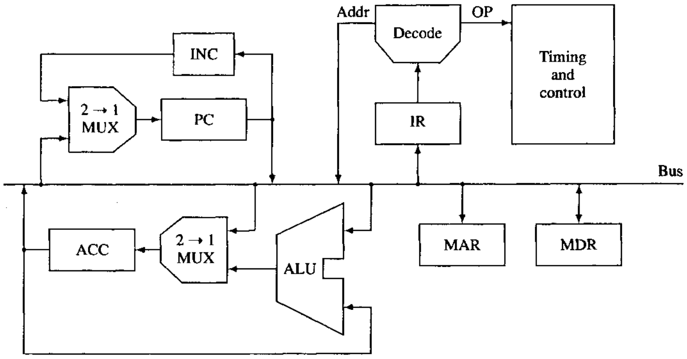
    STORE *addr* — store value in acc. at *addr*

The Accumulator Machine

- This is an *accumulator machine* — it operates only on a value in memory and the value in the accumulator
- The *instruction format* specifies how the instruction is stored internally
  - The whole instruction is 8 bits long
  - 3 bits specify *opcode* — operation to execute
  - 5 bits specify the address; it can address a memory of size 2^5 – 1 = 32 bytes
- Instruction set (2 classes of instructions):
  - Arithmetic — operate on acc. and value in memory, store result in accumulator
  - Load/store — transfer values between acc. and memory (use to fetch first operand, use to store result in variable)

Datapath and Controller for the Accumulator Machine

- The part in upper-right quadrant is the controller; the rest is the data path



Registers	Combinational circuits
ACC (accumulator)	ALU (arithmetic and logic unit)
IR (instruction register)	Decode (instruction decoder)
MAR (memory address register)	INC (incrementer)
MDR (memory data register)	MUX (multiplexer)
PC (program counter)	Timing and control

Diagram from *Computer Systems*, Maccabe, Irwin 1993

Understanding the Accumulator Machine Datapath and Controller

- The components at the lower left should be familiar:
  - ALU — performs arithmetic and logical (none here) operations
    - Gets inputs from accumulator and memory via bus
  - Accumulator (register)
    - Multiplexor selects input from ALU or bus
- The memory system (external to the accumulator machine) connects to the datapath via the bus
  - The accumulator machine accesses the memory through the two registers in the lower right quadrant
  - MAR — memory address register (used to supply address to memory)
  - MDR — memory data register (used to pass data to/from memory)

Understanding the Accum. Machine Datapath and Controller (cont.)

- The datapath components in the upper left quadrant are new
  - PC — program counter — stores the *address* of the next instruction to execute
  - INC — incrementer — can add 1 to the value in the PC
    - To generate address of next instruction (no conditions, loops, or functions yet)
- The components in the upper right quadrant are the datapath's controller
  - IR — instruction register — stores current instruction being executed
    - Same as "state register" in my controller diagram from Lecture 09
  - Decode — instruction decoder, Timing and control
    - These two are the same as the "control logic" and "next state logic" in my controller diagram

Instruction Execution Cycle for the Accumulator Machine

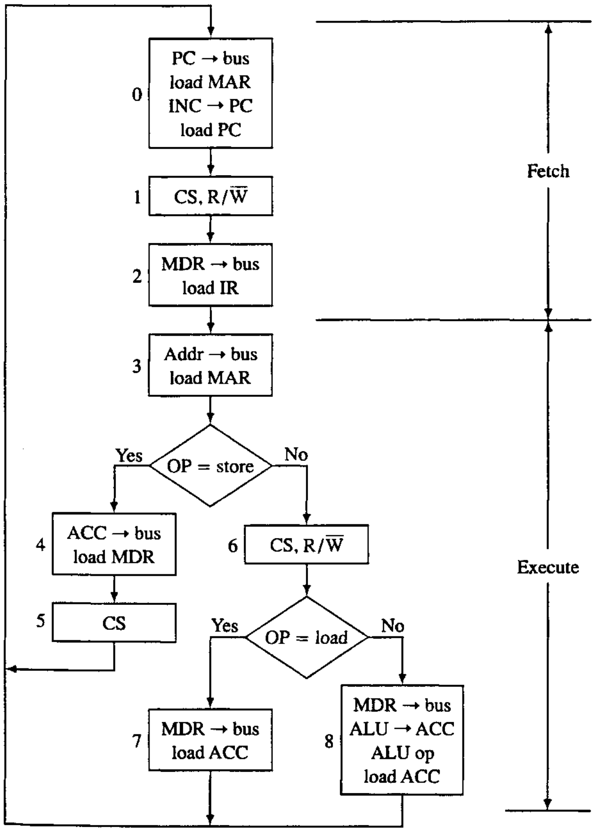
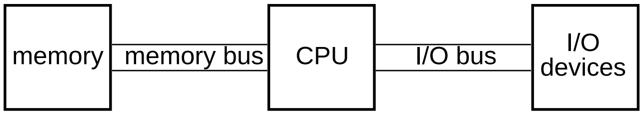


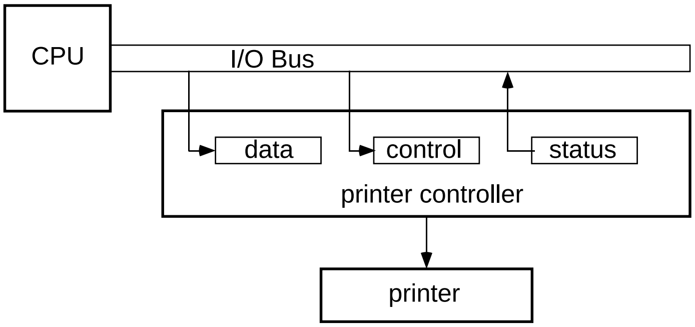
Diagram from *Computer Systems*, Maccabe, Irwin 1993

I/O Devices

- The only component of the computer system left are the I/O devices



- The I/O (input / output) devices are accessed through registers in much the same manner as the memory is accessed



Buses

- In our computer system, there are two kinds of buses
  - Inside the CPU, there is an *internal bus*
  - Between the CPU, memory, and the I/O devices, there is an *external bus*
- A data transfer on an external bus is called a *bus transaction*
  - Usually, one device is the *master* in such a transaction, and the other(s) are the *slaves*
    - When sending data to print, the CPU is the master, and the printer is the slave
    - If there's a print error to report, the printer is the master, and the CPU is the slave
- Briefly scan through the material in Sections 3.4.1 and 3.4.2 of Maccabe, but don't worry about the details

Homework #3 — Due 10/12/98 (Part 2)

2. For each of the following C statements, (i) give a sequence of instructions from Table 3.5 that will evaluate the statement, and then (ii) translate your code sequences into binary using the encoding in Table 3.6 and Figure 3.11. Assume that variables a, b, c, d, and e correspond to memory locations 20, 21, 22, 23, and 24, respectively, and that locations 25 and higher can be used for storage of temporary results.

    a. a = a + b – c

    b. a = bc – ab