Overall organization of computers

Prof Rajeev Barua
EM 003 -- Slide set 3

PLAKSHA UNIVERSITY

What is the CPU?

CPU = Central Processing Unit.

The CPU is the "brain" of the computer.

It has three components:

- Registers
- Arithmetic and Logic Unit (ALU)
- Control unit

CPU chips usually also have part of memory called Cache memory (but not the whole memory).

The CPU’s organization

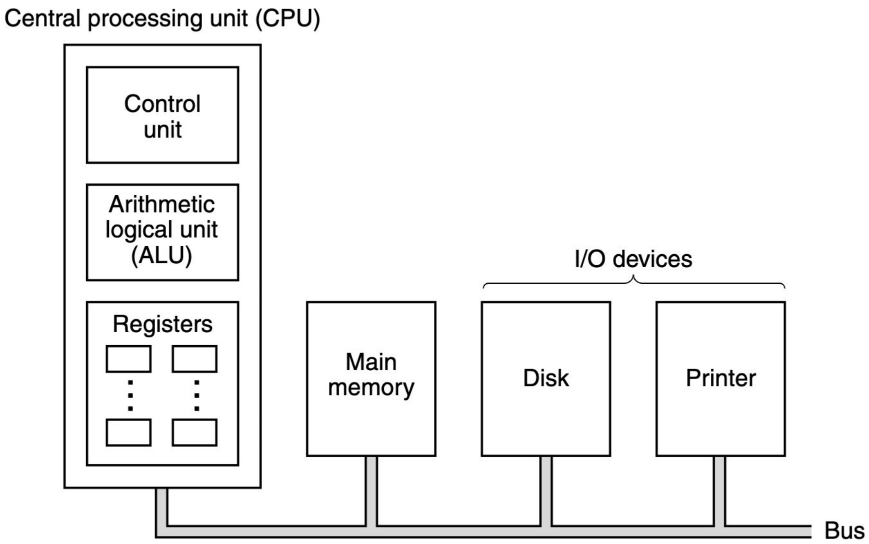


Figure 2-1. The organization of a simple computer with one CPU and two I/O devices.

Why do we have both main memory and disk?

- Other I/O devices include:
  - Printers
  - Monitors
  - Keyboard
  - Mouse
  - Trackpads
  - Joystick
  - CD/DVD drives
  - USB drives
  - Microphone
  - Speakers
  - … (several others)

- The bus is also called the "external bus" or "system bus". (CPU may also have an internal bus).

Why do we have both memory and disk?

Because they have different strengths.

We can combine their best features by using both:

- Just have disk:
  - too slow.
- Just have memory:
  - We lack persistence of storage across turning off.
  - Not enough capacity.
    - memory is much more expensive for same capacity).

Inside the CPU: the data path

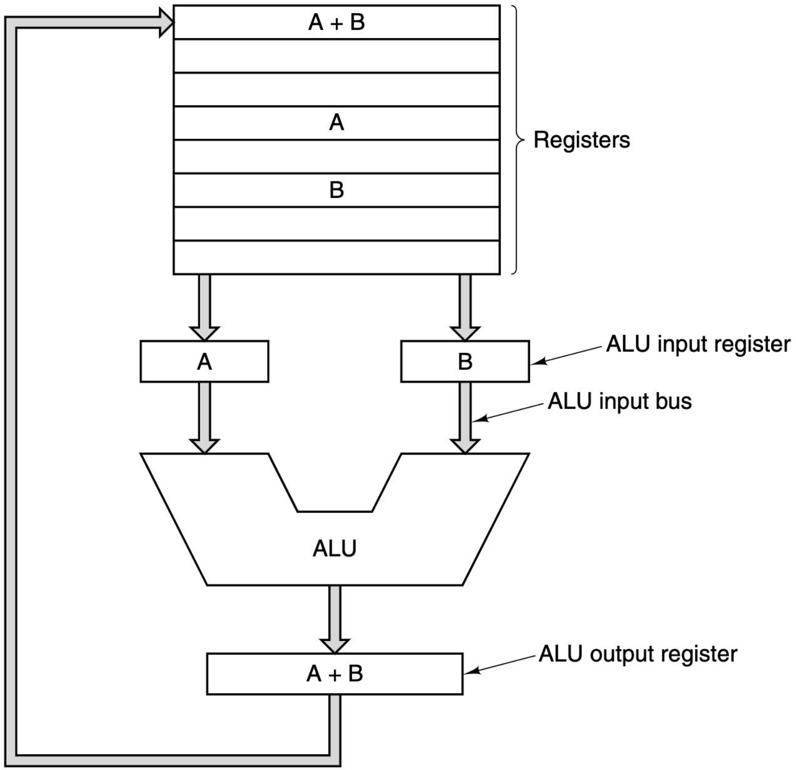


Figure 2-2. The data path of a typical von Neumann machine.

ıs based on select inputs.
om instruction opcode.
' Floating point etc.

d forth between registers and memory)
ı)

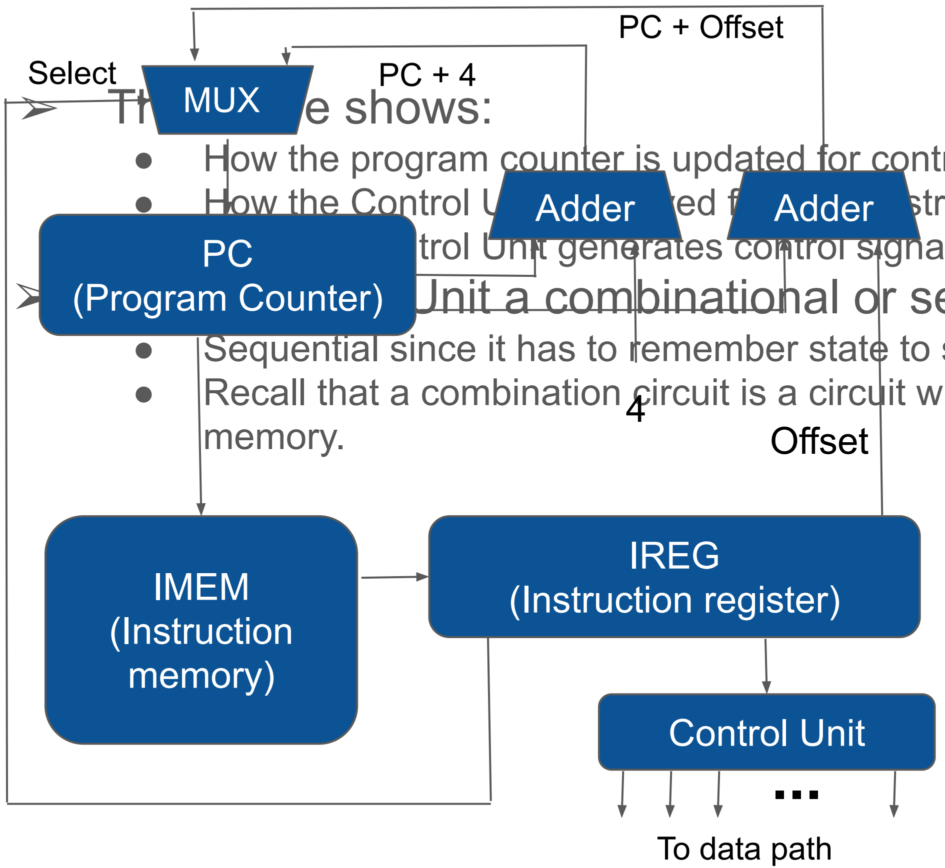
Control Unit and Program Counter integration

Th e shows:

- How the program counter is updated for control flow in programs.
- How the Control U ed f struction encoding.
- trol Unit generates control signals to control the data path.

Unit a combinational or sequential circuit?

- Sequential since it has to remember state to sequence through a sequence of steps.
- Recall that a combination circuit is a circuit without memory, whereas a sequential circuit has memory.



Implementation of Control Unit

Two possible implementations:

- Microprogramming
    - where steps within an instruction are stored as a program.

- Direct execution
    - Where a pipeline controls each instruction in flight by storing IREGs of each stage.

Microprogramming

Microprogramming is where steps within an instruction are stored as a program.

1. Fetch the next instruction from memory into the instruction register.
2. Change the program counter to point to the following instruction.
3. Determine the type of instruction just fetched.
4. If the instruction uses a word in memory, determine where it is.
5. Fetch the word, if needed, into a CPU register.
6. Execute the instruction.
7. Go to step 1 to begin executing the following instruction.

Control signals for each microprogram step are stored in fast "Control store". Usually ROM.

Advantages of interpretation in microprogramming

- Easy to support complex instructions (simply make the microprograms bigger)
- Structured design => Easy debugging of the hardware.
- Could add new instructions even after delivery!

History:

- Trend started around 1950. Reached its heights in 1980 (DEC VAX with more than 200 instructions, and many addressing modes).
- People were concerned with "closing the semantic gap" between hardware and software.
  - ➢ Intuitively appealing, but flawed idea.

Microprogramming is no longer used. Too slow because of extra memory fetch per step, and because it does not work well with pipelining.

Reduced Instruction Set Computing (RISC)

- Around 1980, research started into whether microprogramming could be eliminated.
- Research completed by Patterson [UC Berkeley] and Hennessy [Stanford] in 1985 led to RISC (reduced instruction set computing).

Idea of RISC was:

- simple, fewer instructions (around 50)
- same regular steps for all instructions.
- eliminate sequencing. (just move instruction to next stage every cycle, along with partial results).

Reasons why RISC is faster than microprogramming and CISC:

- Simple regular structure allowed Pipelining.
  - Start next instruction before first is finished!!
- Eliminate one layer of hardware (microprogrammed control).

Modern design principles

These arise from RISC ISAs:

- Pipelining
- Direct execution for control.
- Maximize issue rate
  - ➢ How: simple hardware per stage
- Instructions should be easy to decode
  - ➢ Why: results in simple hardware
- Only load/stores access memory
  - ➢ Why: avoid complicating other instructions that now no longer need to access memory
- Lots of registers
  - ➢ Why: to compensate for slower memory.