Introduction to Computers and Programming

Prof. I. K. Lundqvist

Reading: B pp. 74-98

Lecture 6
Sept 15 2003

Recap

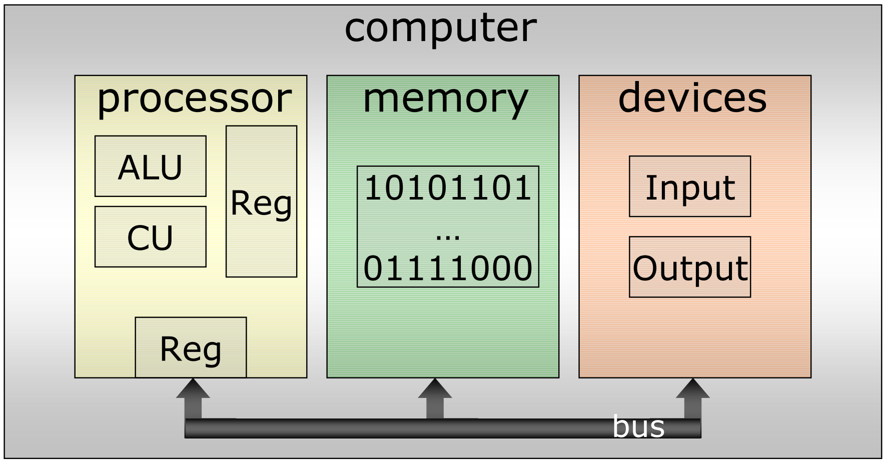
- Number representation
  - Integers
    - The Binary system
    - How to represent negative numbers
  - Real numbers
    - Why floating point numbers
    - How FPs are represented in a computer

How Computers Work

- von Neumann architecture
  - Describes a computer with 4 main sections
    - The Arithmetic Logic Unit (ALU)
    - The Control Unit
    - The memory
    - The input and output devices (I/O)
  - The parts are connected by a bus ("bundle of wires")

Computer Architecture

computer architecture =
computer organization + instruction set architecture



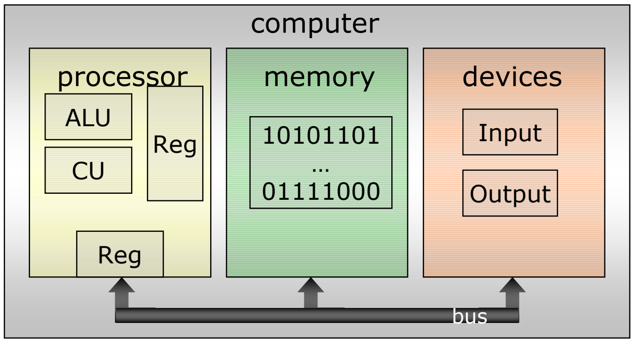
von Neumann architecture

Computer Organization

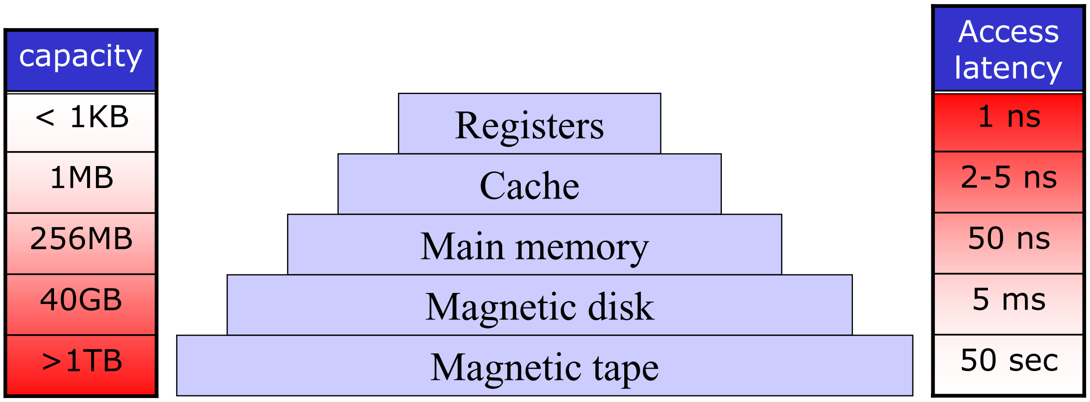
- **CPU**: central processing unit
  - Interprets and carries out the instructions contained in the software
  - **Arithmetic Logic Unit**: performs operations such as addition, subtraction, bit-wise AND, OR, …
  - **Control Unit**: "directs the CPU's operations" fetches instructions from memory, decodes them and produces signals which control the other parts of the computer

Example 1:
Adding values stored in memory

1. Get one of the values to be added from memory and place it in a register
2. Get the other value to be added from memory and place it in another register
3. Activate the addition circuitry with the registers used in 1 and 2 as inputs and another register designated to hold the result
4. Store the result in memory
5. Stop



Storage

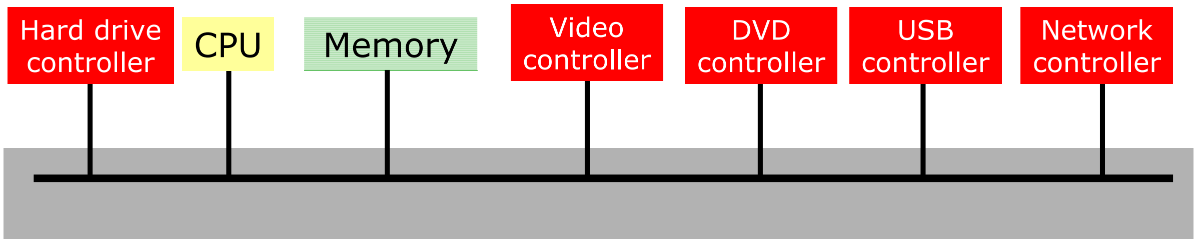


capacity		Access latency
< 1KB	Registers	1 ns
1MB	Cache	2-5 ns
256MB	Main memory	50 ns
40GB	Magnetic disk	5 ms
>1TB	Magnetic tape	50 sec

Input and Output

- Allows the computer to obtain information from the outside world, and send results of its work back again
- Device controllers

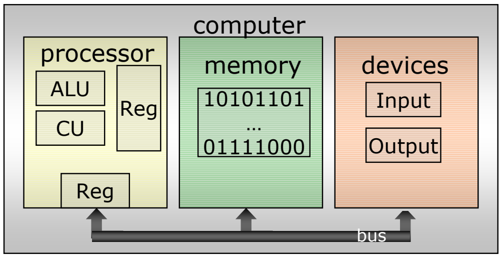
Device Controllers

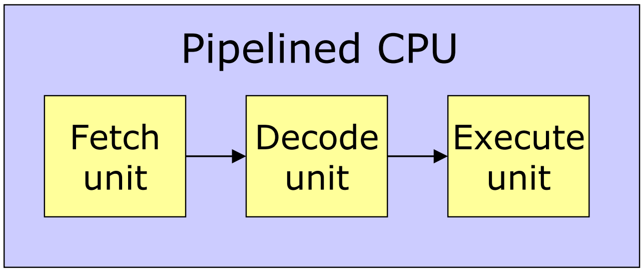


Memory mapped I/O

Direct Memory Access (DMA)

The von Neumann Bottleneck



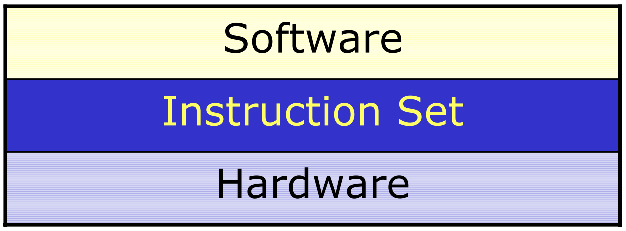


- Cache memory
- Other architectures
  - Pipelined CPU
  - Multiprocessor Machine

Instruction Set Architecture

- **ISA**: spec. detailing the commands a CPU should be able to understand and execute
  - Operations
    - add, sub, mult, …, how is it specified
  - Number of operands
    - 0, 1, 2, 3
  - Operand storage
    - where besides memory
  - Memory address
    - how is memory location specified
  - Type and size of operands
    - byte, integer, float, …
  - …

Instruction Set



Software

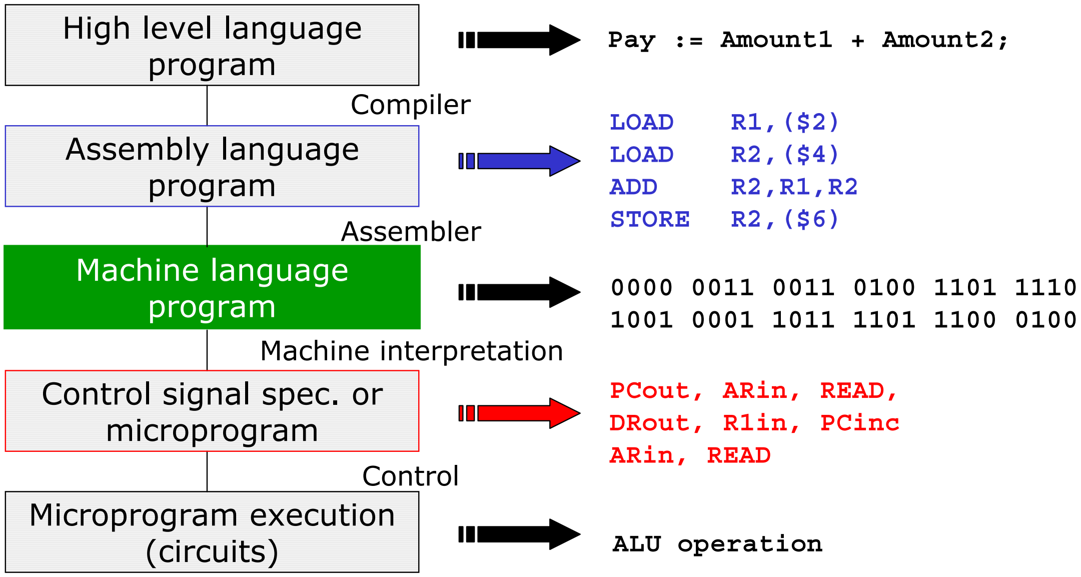
Instruction Set

Hardware

- The collection of *machine language* instructions that a processor understands

- Instructions are bits with well defined fields

Machine Language



High level language program

```
Pay := Amount1 + Amount2;
```

Compiler

Assembly language program

```
LOAD    R1,($2)
LOAD    R2,($4)
ADD     R2,R1,R2
STORE   R2,($6)
```

Assembler

Machine language program

```
0000 0011 0011 0100 1101 1110
1001 0001 1011 1101 1100 0100
```

Machine interpretation

Control signal spec. or microprogram

```
PCout, ARin, READ,
DRout, R1in, PCinc
ARin, READ
```

Control

Microprogram execution (circuits)

```
ALU operation
```

Instruction set

- Instructions in a machine's repertoire can be classified into 3 categories

RISC / CISC

- **Data transfer**
  - LOAD / STORE
  - I/O instructions
- **Arithmetic/logic**
  - Instructions that tell the CU to request an activity within the ALU (+, -, …, XOR, …, SHIFT, ROTATE)
- **Control**
  - Instructions that direct the execution of the program
    - Conditional jumps
    - Unconditional jumps

Dividing values stored in memory

STEP 1. LOAD a register with a value from memory.

STEP 2. LOAD another register with another value from memory.

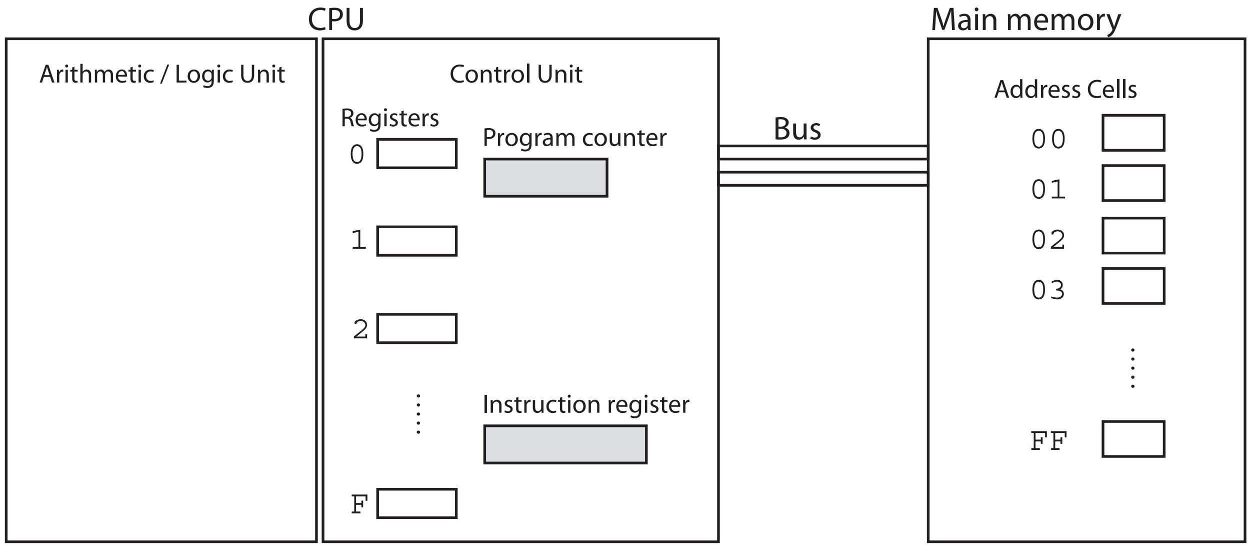
STEP 3. If this second value is zero, JUMP to step 6.

STEP 4. Divide the contents of the first register by the second register and leave the result in a third register.

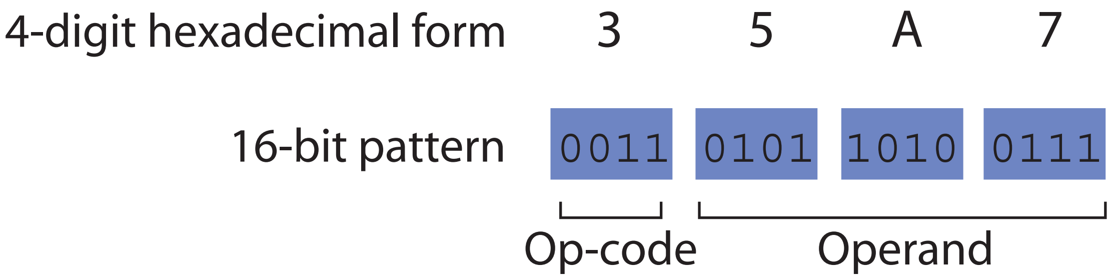
STEP 5. STORE the contents of the third register in memory.

STEP 6. STOP.

The architecture of the machine described in Appendix C



The composition of an instruction for the machine in Appendix C



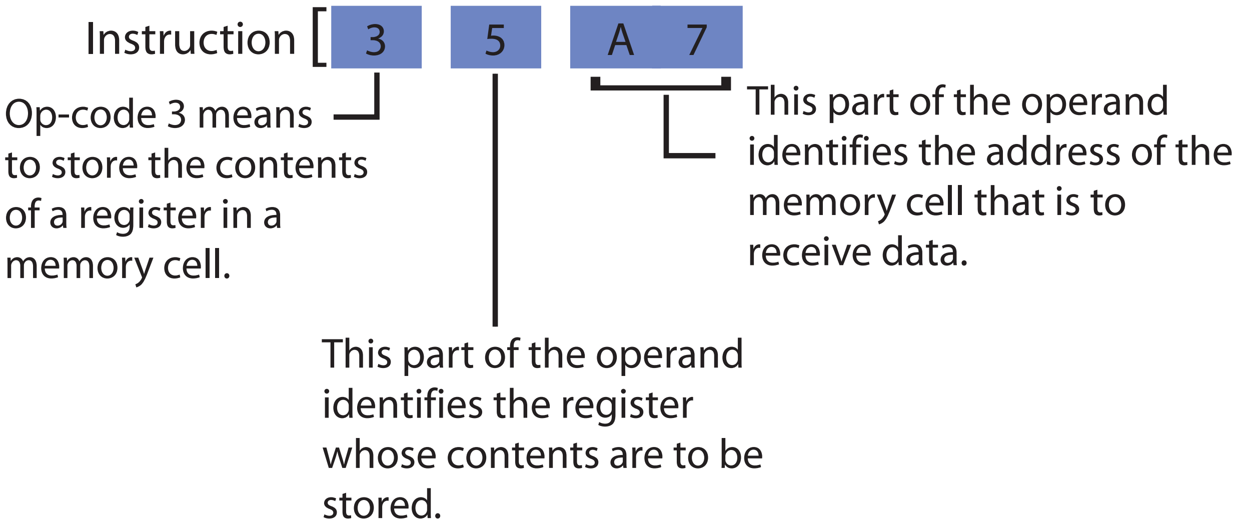
4-digit hexadecimal form 3 5 A 7

16-bit pattern 0011 0101 1010 0111

Op-code

Operand

Decoding the instruction 35A7



Instruction 3 5 A 7

Op-code 3 means to store the contents of a register in a memory cell.

This part of the operand identifies the register whose contents are to be stored.

This part of the operand identifies the address of the memory cell that is to receive data.