Computer Organization and Programming

Prof. Dr. Antônio Augusto Fröhlich
guto@lisha.ufsc.br
http://www.lisha.ufsc.br/~guto

Sep 2006

Outline

- Computer Organization and Programming
  - Data Representation
  - Boolean Algebra
  - Basic Computing System Components
  - Instruction Set Architecture
  - Program translation into machine language (lab)

Numbering Systems

- Decimal System (humans)
  - Base 10

$$123 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$$

$$123.456 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2} + 6 \times 10^{-3}$$

- Binary System (computers)
  - Base 2
  - Represented in digital systems by two voltage levels

$$11001010 = 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 =$$

$$128 + 64 + 8 + 2 = 202$$

Numbering Systems

- Hexadecimal System (assembly)
  - Base 16 (0..9, A, B, C, D, E, F)

$$1234 = 1 \times 16^3 + 2 \times 16^2 + 3 \times 16^1 + 4 \times 16^0 = 4096 + 512 + 48 + 4 = 4660$$

- Hexadecimal <--> Binary
  - Four bits hold an hexadecimal digit

`0x0 = 0000, 0x1 = 0001, ..., 0xf = 1111`

Numbering Systems

- Two's Complement Binary System
  - Base 2
  - Signed integers
    - Most significant bit holds the sign
      - 0 => positive, number stored as ordinary binary
      - 1 => negative, number stored as two's complement
  - Two's complement operation
    - Invert all bits (NOT)
    - Add 1
  - 4-bit

```
0000 = 0, 0001 = 1, 1000 = -8, 1111 = -1
```

Numbering Systems

- Two's Complement Binary System
  - Computers can implement subtraction using the adder circuitry

```
5 - 3 = 5 + -3 = 0101 + 1101 = 0010 = 2
```

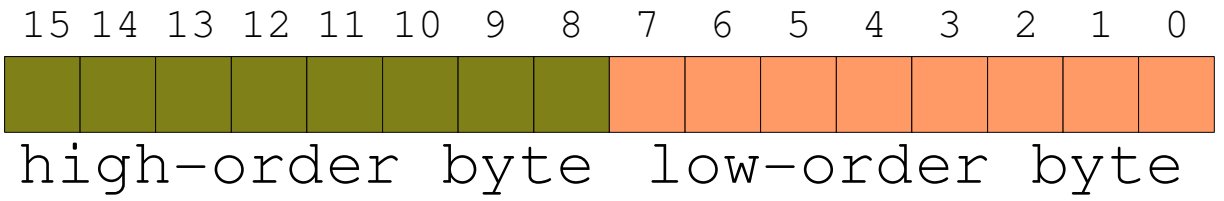
  - Sign extension
    - n-bit number to m-bit number, with m > n
      - Copy the sign bit into the extra bits
        - Semantics is always preserved
    - -3
      - 4-bit => `1101 (0xd)`
      - 8-bit => `11111101 (0xfd)`
      - 16-bit => `1111111111111101 (0xfffd)`

Bits and Bytes

- Bit (Binary digIT)
  - Smallest unit of information in computing
  - A bit has a single binary value, either 0 or 1
  - Bits group to form nibbles (4 bits), bytes (8 bits) and words ($2^{(n >= 3)}$ bits)
- Byte
  - Eight-bit value (today, was not always like that)
  - Bit 0 is the low-order bit or least significant bit
  - Bit 7 is the high-order bit or most significant bit
  - Can hold an ASCII character

Bytes and Words

- Word
  - Natural unit of data in a computer
    - Fixed-size group of bits that are handled together by the machine
    - Influence the size of registers, ULA, memory and I/O buses
    - Typical sizes: 8, 16, 32, 64 bits



Endianness

- Big-endian (Motorola, SPARC, PowerPC)
  - Most significant byte (MSB) is stored at the memory location with the lowest address

`0x12345678 at 100`

99	100	101	102	103	104
	12	34	56	78	

- Little-endian (Intel)
  - Least significant byte (LSB) is stored at the memory location with the lowest address

`0x12345678 at 100`

99	100	101	102	103	104
	78	56	34	12	

Boolean Algebra: Definition

A Boolean algebra is a set A, supplied with two binary operations $\wedge$ (AND), $\vee$ (OR), a unary operation $\neg$ (NOT) and two elements 0 (zero) and 1 (one), such that, for all elements a, b and c of set A, the following axioms hold:

Boolean Algebra: Axioms

Associativity

$a \vee (b \vee c) = (a \vee b) \vee c$ $\qquad$ $a \wedge (b \wedge c) = (a \wedge b) \wedge c$

Commutativity

$a \vee b = b \vee a$ $\qquad$ $a \wedge b = b \wedge a$

Absorption

$a \vee (a \wedge b) = a$ $\qquad$ $a \wedge (a \vee b) = a$

Distributivity

$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ $\qquad$ $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$

Complements

Boolean Algebra: Identities

Idempotency

$a \vee a = a$ $\qquad$ $a \wedge a = a$

Boundedness

$a \vee 0 = a$ $\qquad$ $a \wedge 1 = a$

$a \vee 1 = 1$ $\qquad$ $a \wedge 0 = 0$

Complements

$\neg 0 = 1$ $\qquad$ $\neg 1 = 0$

De Morgan's laws

$\neg (a \vee b) = \neg a \wedge \neg b$ $\qquad$ $\neg (a \wedge b) = \neg a \vee \neg b$

Involution

$\neg\neg a = a$

Truth Tables

- Can represent boolean functions
  - OR

OR	0	1
0	0	1
1	1	1

  - AND

AND	0	1
0	0	0
1	0	1

  - Four-varible function

F = AB + CD		BA			
		00	**01**	**10**	**11**
DC	**00**	0	0	0	1
	01	0	0	0	1
	10	0	0	0	1
	11	1	1	1	1

Truth Tables

- Alternative representation

F = AB + C		BA			
		00	01	10	11
C	0	0	0	0	1
	1	1	1	1	1

Truth Tables

- Alternative representation

C	B	A	F = ABC	F = AB + C	F = A + BC
0	**0**	**0**	0	0	0
0	**0**	**1**	0	0	1
0	**1**	**0**	0	0	0
0	**1**	**1**	0	1	1
1	**0**	**0**	0	1	0
1	**0**	**1**	0	1	1
1	**1**	**0**	0	1	1
1	**1**	**1**	1	1	1

Canonical Forms

- Allows for function simplification
  - For any given boolean function there exists a unique canonical form
- Disjunction of minterms (sums of products)
  - Minterm is a logical expression of n variables consisting of only the logical conjunction operator and the complement operator
    - Three variables: `A'B'C', AB'C', A'BC', ABC', A'B'C, AB'C, A'BC, ABC`

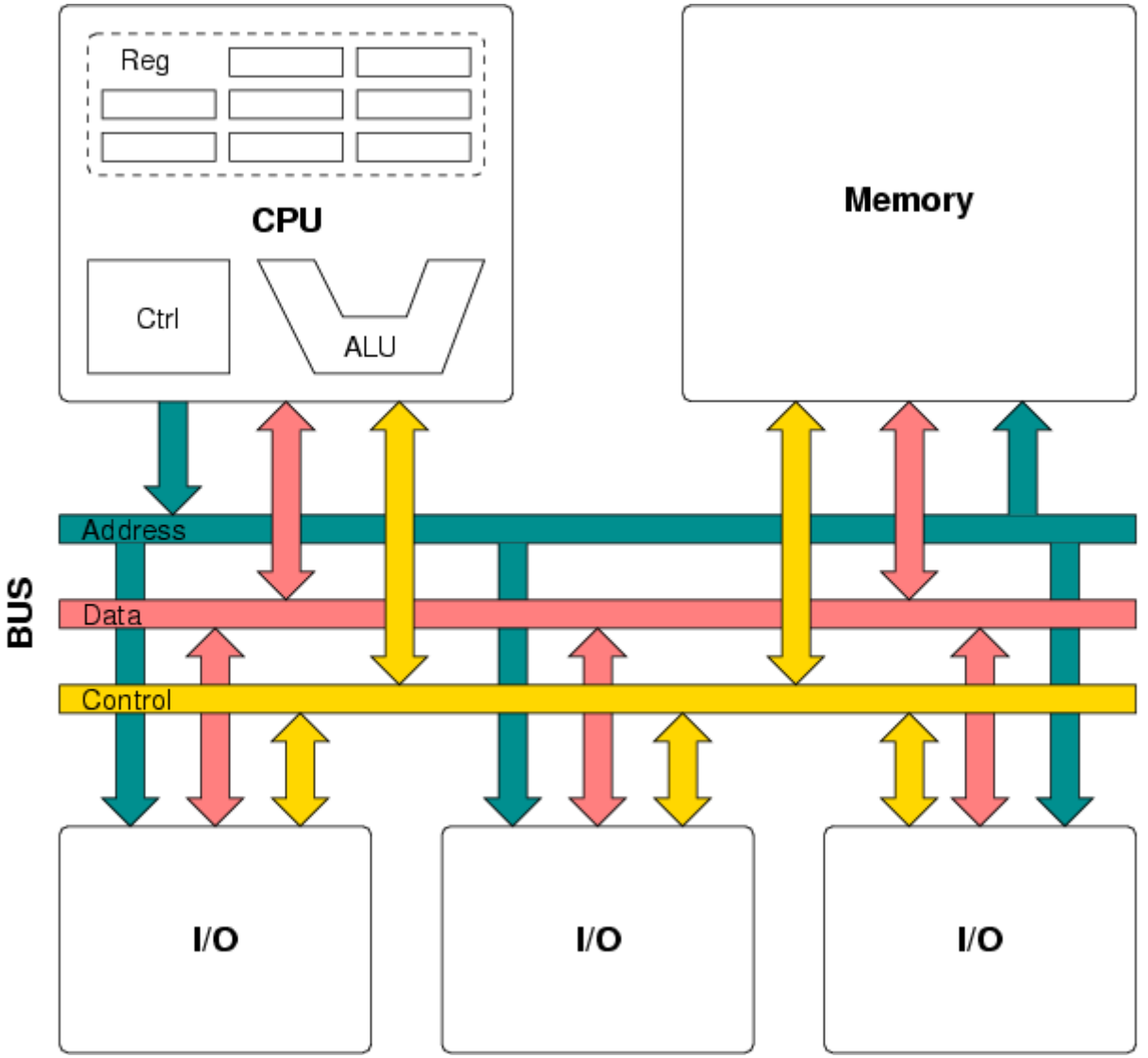
`F = AB + C = ABC+A'BC+AB'C+A'B'C+ABC'`

Canonical Forms

- Conjunction of maxterms (product of sums)
  - Maxterm is a logical expression of n variables consisting of only the logical disjunction operator and the complement operator
    - Three variables: `A'+B'+C', A+B'+C', A'+B+C', A+B+C', A'+B'+C, A+B'+C, A'+B+C, A+B+C`

`F = AB + C = (A+B+C) (A'+B+C) (A+B'+C)`

Basic Computer Organization



Central Processing Unit

- Processor
  - Interprets instructions and operates on data
- Elements
  - Control Unit
    - Controls the CPU, fetches instructions and data for the ALU
  - Arithmetic and Logic Unit (ALU)
    - Perform arithmetic and logic operations on data stored in registers
  - Registers
    - Holds data being manipulated by the ALU

Main Memory

- Stores data and instructions
  - Von Neumann: single memory
  - Harvard: separate memories
- Read/write access
- Usually byte-addressed (not word-addressed anymore)
- Technologies
  - RAM, ROM, EEPROM, Flash

I/O Devices

- Connect the computer to the “world”
- Devices are usually connected through secondary buses (i.e. ISA, PCI, VME, USB)
- Input
  - Keyboard, mouse
- Output
  - Monitor, printer
- Storage
  - Disk, DVD, flash
- Network
  - Ethernet, Wi-Fi

System Bus

- Interconnect system components
  - CPU
  - Memory
  - I/O devices
- Signals
  - Addresses
  - Data
  - Control
    - R/W, I/O, INT, REQ, GRANT, CLOCK

Instruction Set Architecture

"An instruction set, or instruction set architecture (ISA), is the part of the computer architecture related to programming, including the native data types, instructions, registers, addressing modes, memory architecture, interrupt and exception handling, and external I/O. An ISA includes a specification of the set of opcodes (machine language), the native commands implemented by a particular CPU design."

(Wikipedia)

Instruction Sets X CPU Storage

- Internal storage in the CPU
  - Stack (operands on the top of the stack)
  - Accumulator (single accumulator “register”)
  - Set of registers

C = A + B

Stack	Accumulator	Register
`push A`	`load A`	`load r1, A`
`push B`	`add B`	`add r1, B`
`add`	`store C`	`store C, r1`
`pop C`		

Register Machines

- Number of operands for ALU instructions

```
add r1, r2        (r1 <- r1 + r2)
add r1, r2, r3    (r1 <- r2 + r3)
```

- Number of memory operands for ALU instructions
  - Zero (register-register)

    ```
    add r1, r2    (r1 <- r1 + r2)
    ```

  - One (register-memory)

    ```
    add r1, B     (r1 <- r1 + B)
    ```

  - Many (memory-memory)

    ```
    add C, A, B   (C <- A + B)
    ```