

### What will you learn?

- Programs translated into machine learning
- Hardware -> software interface
- Parallel processing

### Seven Great ideas

- abstraction
  - common case fast
  - pipelining
  - prediction
  - Parallelism
  - Hierarchy of memories
  - Dependability
- } Performance

### Grading

- Evaluate research papers
  - Conduct a research survey
  - Written exam (55%)
- } 45%

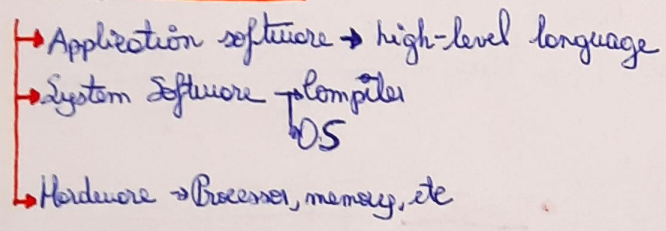
## T1

### Fundamentals of Computer Architectures

#### Computer Revolution

Computers went from old useless machines to smartphones

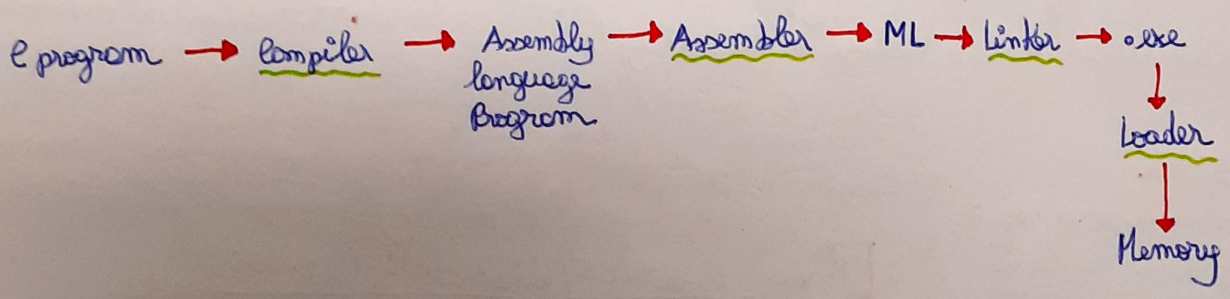
#### Below your program



#### Levels of Program Code

- High-level language (ex: C, Java)
- Assembly language
- Hardware Representation (binary digits)

#### Translating and Starting a program





## Components of a Computer

- Processor (gets data from memory)
- Memory
- Control (sends the signals)

## Performance

- Algorithm (n° of operations executed)
- Programming language, compiler, architecture
- Processor and memory system
- I/O system (including OS)

Response Time → how long it takes to do a task

Throughput → Total work done per hour unit

$$\text{Performance} = \frac{1}{\text{Execution Time}}$$

## Measuring Execution Time

- Elapsed time → total response time
- CPU time → time spent processing a job

## CPU Clocking

- Clock period (duration of a clock cycle)
- Clock frequency (rate): cycles per second
- CPU Time = CPU clock cycles × Clock cycle time =  $\frac{\text{CPU clock cycles}}{\text{Clock rate}}$

## CPU performance equation

- Clock cycle = Instruction count × Cycles per Instruction
- CPU Time = Instruction count × CPI × Clock cycle Time =  $\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock Rate}}$

MIPS → Millions of instructions per second

$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

$$\frac{\text{Execution time}}{\text{time}} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock Rate}}$$

## Performance Summary

$$\text{CPU time} = \frac{\text{Instructions}}{\text{Programs}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

$$\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

## Performance depends

- Algorithm
- Programming
- Compiler
- Instruction set

## Reducing power

• Suppose a main cpu has:

- 85% of capacity load
- 15% of voltage reduction
- 15% of frequency reduction

$$\frac{P_{\text{new}}}{P_{\text{old}}} = \frac{P_{\text{old}} \cdot 0.85 (V_{\text{old}} \cdot 0.85)^2 (F_{\text{old}} \cdot 0.85)}{P_{\text{old}} \cdot V_{\text{old}}^2 \cdot F_{\text{old}}} = 0.52$$

↓  
half the



## Multiprocessors

- execute multiple instructions
- harder to programming for performance load balancing

## Pitfall: Amdahl's Law

$$T_{\text{improved}} = \frac{T_{\text{affected}}}{\text{improvement factor}} + T_{\text{unaffected}}$$

## T2]

### Data Representation

- Most computers use blocks of eight bits, or bytes, as the smallest addressable unit of memory

### Stored Computer Program concept

- Program → Memory
  - Accounting Program
  - Editor program
  - C compiler
  - Payroll data
  - Book text
  - Source code in C

} Machine code

## Using bits to represent numbers

- Binary: 0000 0000 to 1111 1111
- Decimal: 0 to 255
- Hexadecimal: 00 to FF

### conversion table

Hex	Dec	Bin
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

## Unsigned Binary Integers

- $X = X_{n-1}2^{n-1} + X_{n-2}2^{n-2} + \dots + X_02^0$
- Range: 0 to  $2^n - 1$

## 2's - Complement Signed Integers

- $X = X_{n-1}2^{n-1} + X_{n-2}2^{n-2} + \dots + X_02^0$
- Range:  $-2^{n-1}$  to  $2^{n-1} - 1$

- Most significant bit is a sign bit
- Complement and add 1

$$\begin{aligned} X + \bar{X} &= 1111\dots 11_2 = -1 \\ \bar{X} + 1 &= -X \end{aligned}$$

## Data representations

- limited numbers of bits
- operations can underflow/overflow
- everything is represented by numbers only

## Data table

C data type	bits
char	1
short	2
int	4
long	4
long long	8
float	4
double	8
long double	10/12
pointer	4

## Using Bits to represent code

- A program is a sequence of bytes
- Different machine types (Windows/Linux, etc) use different and incompatible instructions and encodings

## Boolean Algebra

- True = 1
- False = 0

• And  $\rightarrow A \& B$

$\Phi$	0	1
0	0	0
1	0	1

• Not  $\rightarrow \neg A$

$\neg$
0
1

• Or  $\rightarrow A | B$

1	0	1
0	0	1
1	1	1

• Xor  $\rightarrow A \wedge B$

1	0	1
0	0	1
1	1	0



## Operate on bit vectors (Boolean)

- $\&$   $\rightarrow$  Intersection
- $|$   $\rightarrow$  Union
- $\wedge$   $\rightarrow$  Symmetric difference
- $\sim$   $\rightarrow$  Complement

ex: 
$$\begin{array}{r} 0110\ 1001 \\ \& 0101\ 0101 \\ \hline 0100\ 0001 \end{array}$$

$$\begin{array}{r} 0110\ 1001 \\ | 0101\ 0101 \\ \hline 0111\ 1101 \end{array}$$

$$\begin{array}{r} 0110\ 1001 \\ \wedge 0101\ 0101 \\ \hline 0011\ 1100 \end{array}$$

$$\begin{array}{r} 0101\ 0101 \\ \sim 0101\ 0101 \\ \hline 1010\ 1010 \end{array}$$

## Bit level Operations

- implement masking operations
- Very useful in:
  - $\rightarrow$  Hash tables
  - $\rightarrow$  Controlling devices
  - $\rightarrow$  IP addressing
  - $\rightarrow$  Image processing

## Shift Operations

- shifting bit patterns to the left and to the right
- Left Shift:  $x \ll y$ 
  - $\rightarrow$  Fill with 0's on right
- Right Shift:  $x \gg y$ 
  - $\rightarrow$  Fill with 0's on left
  - $\rightarrow$  Replicate most significant bit on left

## Machine Words

- 32-bit  $\rightarrow$  limit addresses to 4 GB ( $2^{32}$  bytes)
- 64-bit  $\rightarrow$  limit addresses to 16 PB ( $2^{64}$  bytes)

## Pointers in C

- Pointer is a reference to another variable
- Pointer size is determined by word-size of machine
  - $\rightarrow$  32 bit architecture - 4 bytes
  - $\rightarrow$  64 bit architecture - 8 bytes

## Byte Ordering

- multi-byte word ordered in memory:
  - $\rightarrow$  Big Endian - least significant byte has highest address
    - Used in Internet
  - $\rightarrow$  Little Endian - least significant byte has lowest address
    - Used by x86, Android, iOS

ex:

0x01234567

Big | 01 23 45 67

Little | 67 45 23 01