

CS2100

<http://www.comp.nus.edu.sg/~cs2100/>

COMPUTER ORGANISATION

Lecture #1

Introduction

From High-Level Languages to Computer Organisation
(AY2019/20 Semester 2)



NUS
National University
of Singapore

School of
Computing

Lecture #1: Introduction

1. Programming Languages
2. C Programming Language
3. Abstraction
4. So, What is a Computer?
5. Why Study Computer Organisation?

1. Programming Languages (1/5)

Programming language: a formal language that specifies a set of instructions for a computer to implement specific algorithms to solve problems.



1. Programming Languages (2/5)



High-level program

Eg: C (CS1010), Java (CS1010J),
Python (CS1010S), ECMAScript
(CS1101S)

```
int i, a = 0;
for (i=1; i<=10; i++) {
    a = a + i*i;
}
```

```
a = 0
for i in range(1,11):
    a = a + i*i
```

Low-level program

Eg: MIPS (CS2100)

```
addi $t1, $zero, 10
add  $t1, $t1, $t1
addi $t2, $zero, 10
Loop: addi $t2, $t2, 10
      addi $t1, $t1, -1
      beq  $t1, $zero, Loop
```

Machine code

Computers can
execute only machine
code directly.

```
001000000000010010000000000001010
00000001001010010100100000100000
. . .
```

1. Programming Languages (3/5)



❖ 1st Generation Languages

Machine language.
Directly executable by machine.
Machine dependent.
Efficient code but difficult to write.

❖ 2nd Generation Languages

Assembly language.
Need to be translated (**assembled**) into machine code for execution.
Efficient code, easier to write than machine code.

❖ 3rd Generation Languages

Closer to English.
Need to be translated (**compiled** or **interpreted**) into machine code for execution.
Eg: FORTRAN, COBOL, C, BASIC

❖ 4th Generation Languages

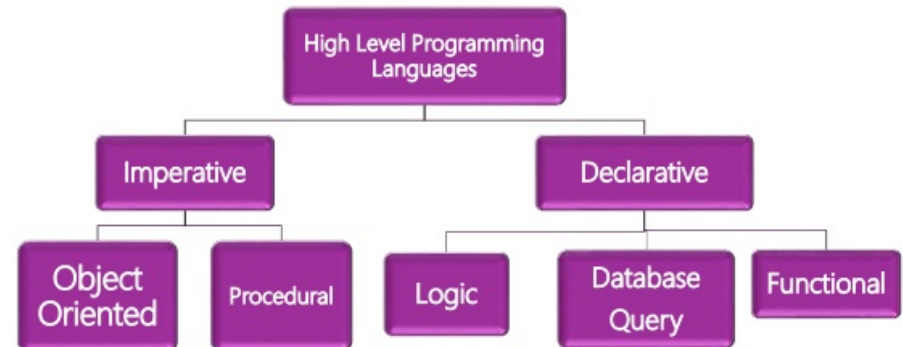
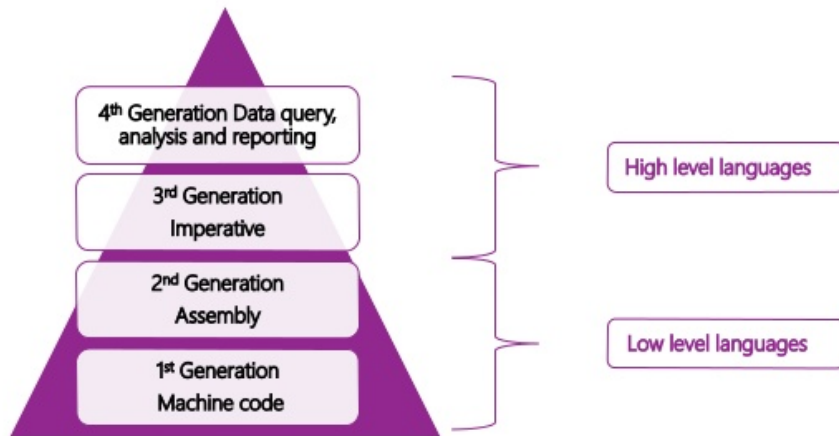
Require fewer instructions than 3GL.
Used with databases (query languages, report generators, forms designers)
Eg: SQL, PostScript, Mathematica

❖ 5th Generation Languages

Used mainly in A.I. research.
Declarative languages
Functional languages (eg: Lisp, Scheme, SML)
Logic programming (eg: Prolog)

[illegible]

- # paradigm.



1. Programming Languages (5/5)



- Difference between **scripting languages** and **programming languages**

Scripting languages	Programming languages
Eg: JavaScript, PHP, Python	Eg: C, C++, Java
Interpreted; do not require compilation	Compiled
Generally slower	Generally faster

- However, the environment is more important than the language in the classification.
 - We can write a C interpreter and use it as a scripting language
 - We can compile JavaScript to machine code
- The distinction is getting blurred due to improved hardware capabilities and coding practices
 - Eg: Python is widely used without compilation, but the main implementation (CPython) does that by compiling to bytecode on-the-fly and then running the bytecode in a VM (virtual machine)
 - Eg: Java is compiled to bytecode, which is then interpreted/recompiled at runtime

2. C Programming Language (1/4)

- Created by Dennis Ritchie (1941 – 2011) at Bell Laboratories in the early 1970s.
- C is an **imperative procedural language**.
- C provides constructs that map efficiently to typical machine instructions.
- C is a high-level language very close to the machine level, hence sometimes it is called “mid-level”.
- UNIX is written in C.

Dennis Ritchie
1941-2011



HelloWorld.py

```
print("Hello, world")
```

HelloWorld.c

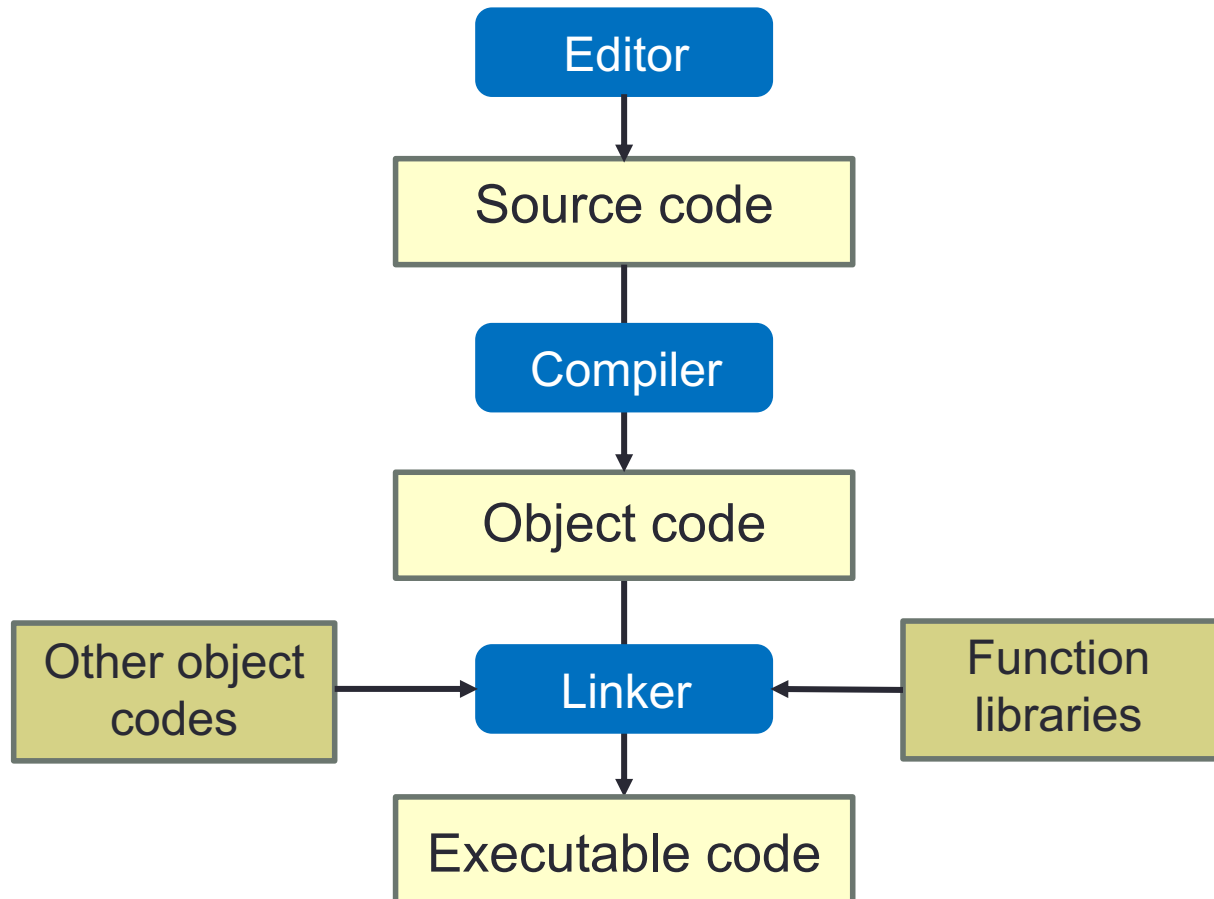
```
#include <stdio.h>

int main(void) {
    printf("Hello, world\n");
    return 0;
}
```

(Note: All C programs in the lectures are available on LumiNUS as well as the CS2100 website. Python versions are also available.)

2. C Programming Language (2/4)

- Creating a C program



2. C Programming Language (3/4)

[Edit](#)

```
tantc@suna0:~/cs2100/lect/prog/lect1[40]: vim HelloWorld.c
```

Illustration on SoC
UNIX server

```
1 // HelloWorld.c
2 #include <stdio.h>
3
4 int main(void) {
5     printf("Hello, world\n");
6     return 0;
7 }
8
~
~
```

[Compile and link](#)

```
tantc@suna0:~/cs2100/lect/prog/lect1[42]: ls
HelloWorld.c
tantc@suna0:~/cs2100/lect/prog/lect1[43]: gcc HelloWorld.c
tantc@suna0:~/cs2100/lect/prog/lect1[44]: ls
a.out      HelloWorld.c
tantc@suna0:~/cs2100/lect/prog/lect1[45]:
```

[Execute](#)

```
tantc@suna0:~/cs2100/lect/prog/lect1[46]: a.out
Hello, world
tantc@suna0:~/cs2100/lect/prog/lect1[47]:
```

2. C Programming Language (4/4)

```
tantc@suna0:~/cs2100/lect/prog/lect1[43]: gcc HelloWorld.c  
tantc@suna0:~/cs2100/lect/prog/lect1[44]: ls  
a.out          HelloWorld.c  
tantc@suna0:~/cs2100/lect/prog/lect1[45]:
```

Compile
and link

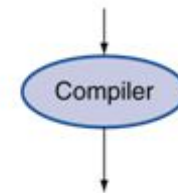
- The command **gcc** hides all the details
- Using **gcc -v HelloWorld.c** will display all the details
- The process goes through the following steps to generate machine code:
 - Preprocessing
 - Compilation
 - Assembler
 - Linker

3. Abstraction (1/3)

- High-level language
 - Level of abstraction closer to problem domain
 - Provides productivity and portability
- Assembly language
 - Textual and symbolic representation of instructions
- Machine code (object code or binary)
 - Binary bits of instructions and data

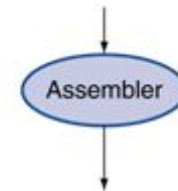
High-level
language
program
(in C)

```
swap(int v[], int k)
{
    int temp;
    temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
}
```



Assembly
language
program
(for MIPS)

```
swap:
    muli $2, $5, 4
    add $2, $4, $2
    lw $15, 0($2)
    lw $16, 4($2)
    sw $16, 0($2)
    sw $15, 4($2)
    jr $31
```

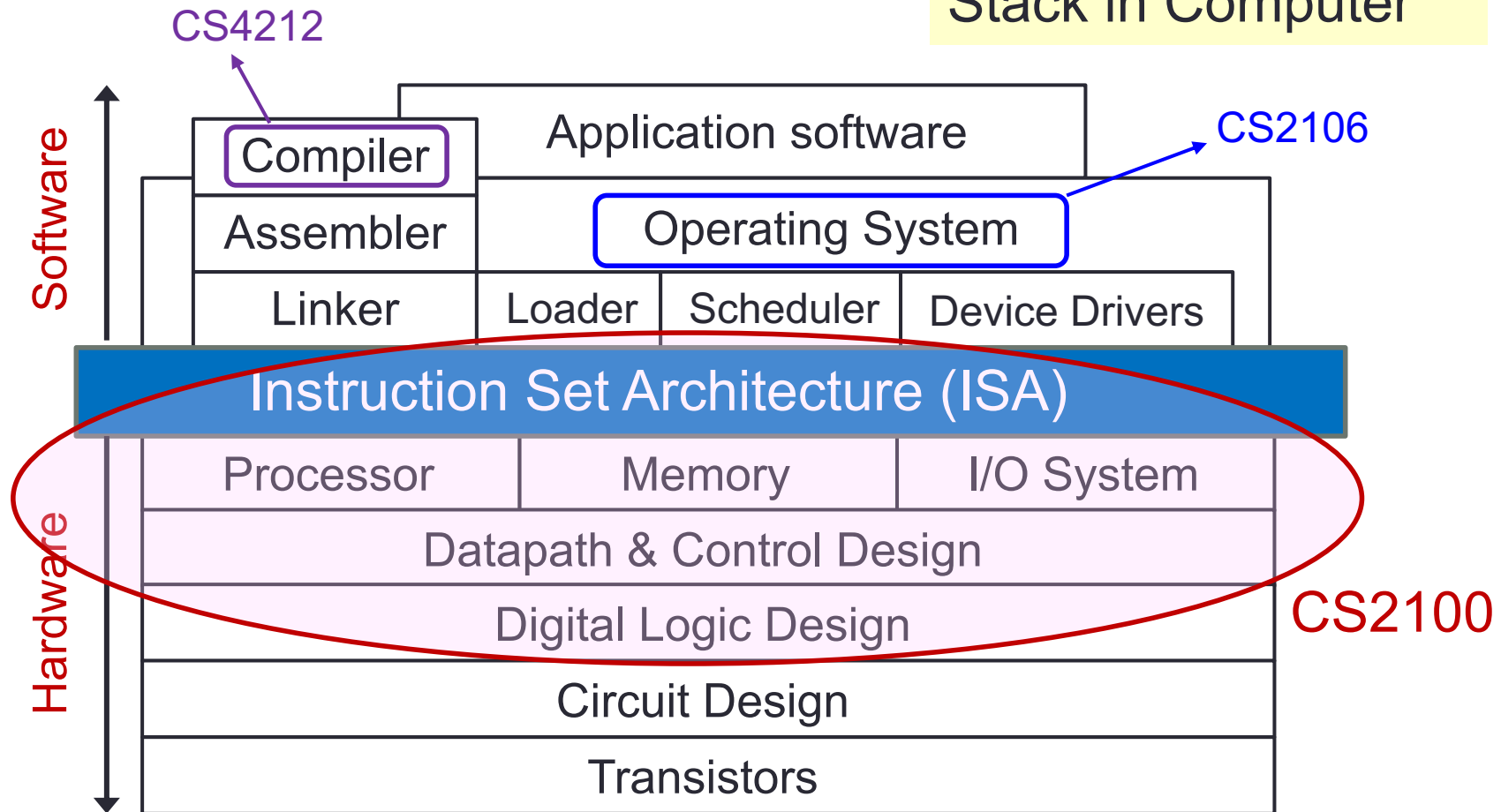


Binary machine
language
program
(for MIPS)

```
000000001010000100000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
100011001111001000000000000000100
101011001111001000000000000000000
101011000110001000000000000000100
00000011111000000000000000001000
```

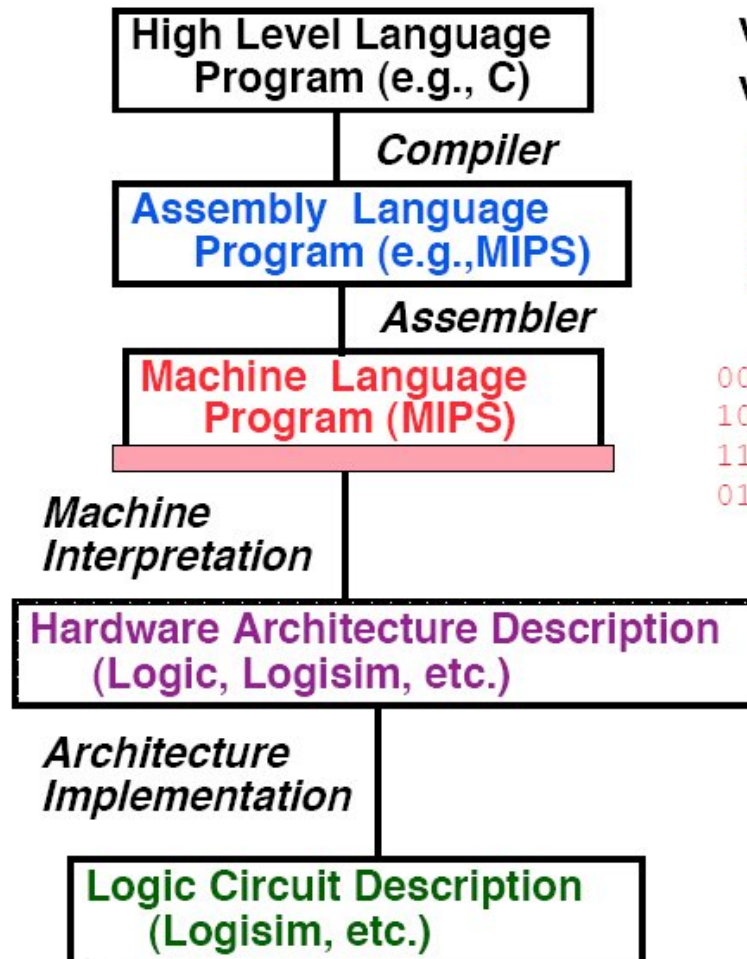
3. Abstraction Layers (2/3)

Hardware/Software Stack in Computer



3. Abstraction (3/3)

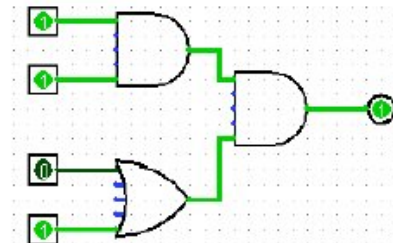
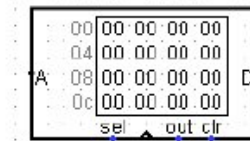
Level of Representation



```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

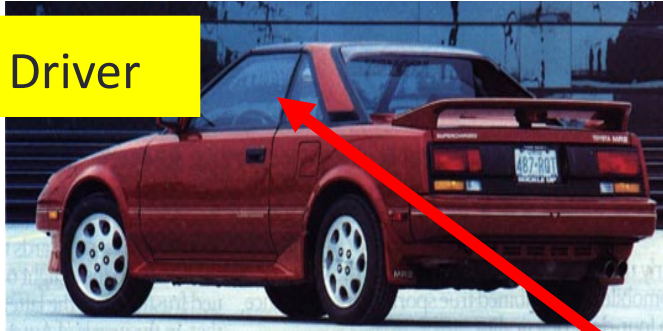
```
lw $t0, 0($2)
lw $t1, 4($2)
sw $t1, 0($2)
sw $t0, 4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```



4. So, What is a Computer? (1/6)

Driver



Example: An automobile augments our power of locomotion.

A computer is a device capable of solving problems according to designed programs. It simply augments our power of storage and speed of calculation.



Programmer

4. So, What is a Computer? (2/6)

- From **computer organisation** perspective, we study the **components** and **how they work together**
 - Processor, memory, input/output devices, networks, ...



Credit: <http://tech4abc.blogspot.sg/2010/08/latest-technology-in-computer-hardwares.html>

4. So, What is a Computer? (3/6)

MOST RECENT July 8th, 2012 1 COMMENT »

Most Recent Computer Technology

Written by: admin
Tags: [breaking](#)



Do you know what is in your computer? Maybe you peeked when the repair technician was installing amazing for you. When you primary open up the CPU and seem inside, a computer is a very intimidating machine. But

once you are acquainted with about the dissimilar parts that make up a total computer it gets a lot easier. Today's computer consists of around eight main devices; some of the advanced computers might have a few additional mechanisms. What are these eight main components and what are they used for? We will start with beginner level facts to get you in progress.

First is the Power Supply. The authority provides is used to provide electrical

SHARE

0

Like

Digg

1. Power supply
2. Motherboard
3. Central Processing Unit (CPU)
4. Random Access Memory (RAM)
5. Hard drive
6. Cooling fan
7. I/O devices



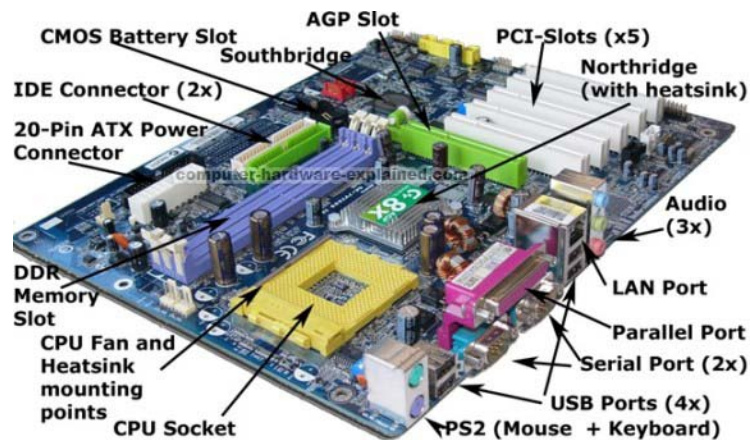
Credit:

http://www.overclock3d.net/reviews/cpu_mainboard/the_computer_council_-_clocked_gamer_quad/1

Credit: <http://tech3news.com/most-recent-computer-technology/>

4. So, What is a Computer? (4/6)

■ PC motherboard

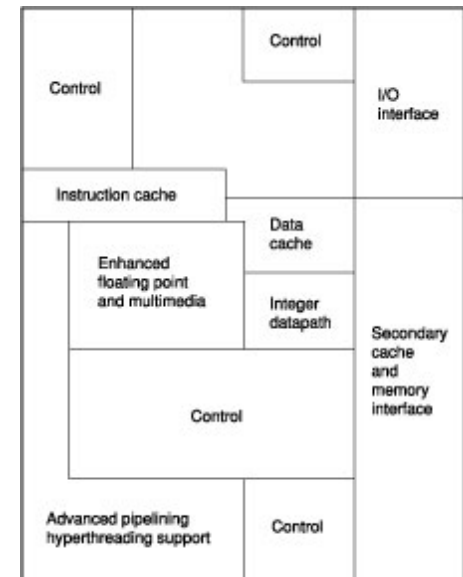
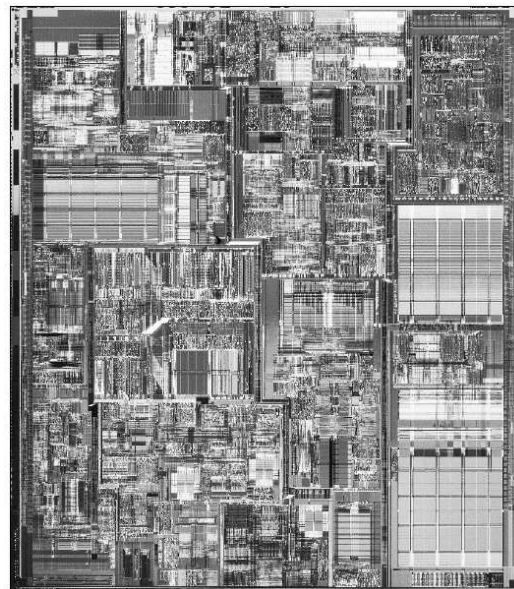


Credit: <http://www.computer-hardware-explained.com/what-is-a-motherboard.html>

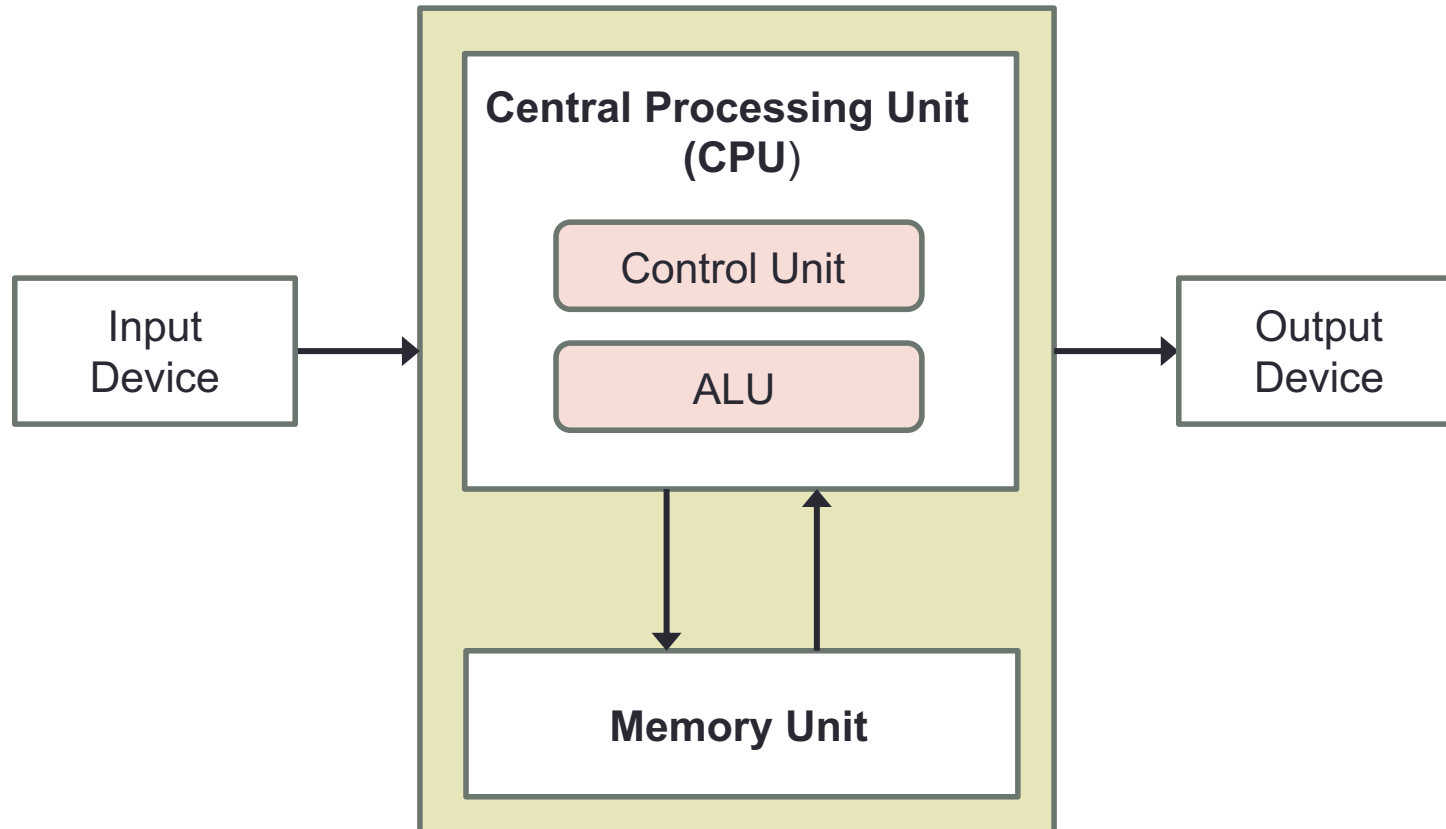
■ Pentium processor



Inside a Pentium chip



4. So, What is a Computer? (5/6)



ALU: Arithmetic/Logic Unit

4. So, What is a Computer? (6/6)

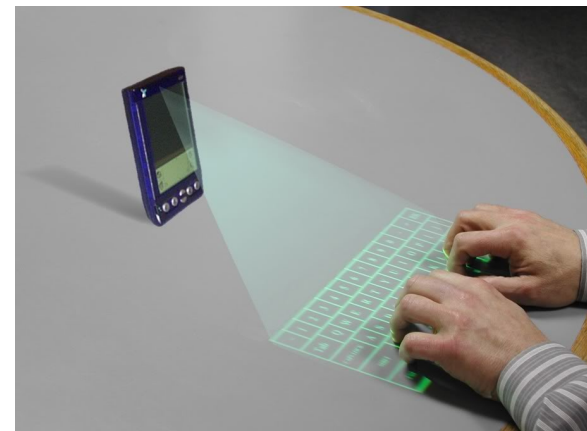
Next generation...



Credit: <http://www.prabhanjamindiaits.com/blogdetailedpage.aspx?id=66>



Credit: <http://www.custom-build-computers.com/Latest-Computer-Hardware.html>



Credit: <http://new-techpc.blogspot.sg/2012/10/latest-in-computer-technology.html>

6th January 2014

www.theverge.com/2014/1/6/5282472/intel-announces-edison-a-computer-the-size-of-an-sd-card



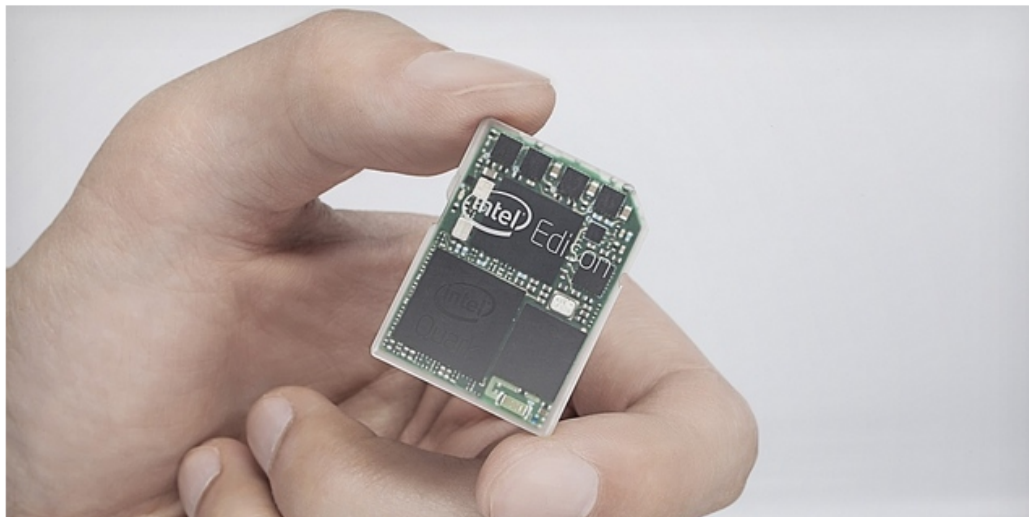
LONGFORM VIDEO REVIEWS TECH SCIENCE CULTURE DESIGN BUSINESS US & WORLD FOR

CES 2014 TECH BREAKING

Intel announces Edison, a computer the size of an SD card

By **Sean Hollister** on January 6, 2014 10:01 pm [Email](#)

DON'T MISS STORIES *FOLLOW THE VERGE* [g+](#) [f Like](#) 280k [t Follow](#) 355K followers

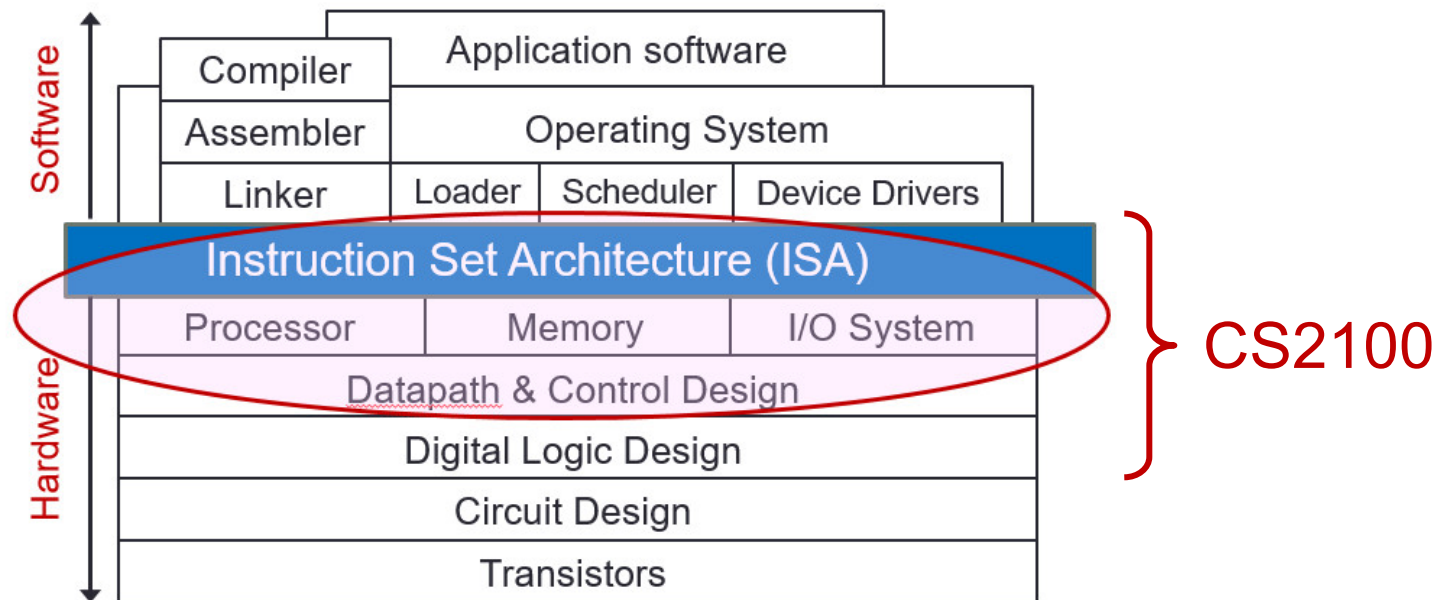


<http://www.theverge.com/2014/1/6/5282472/intel-announces-edison-a-computer-the-size-of-an-sd-card>



5. Why Study Computer Organisation?

- **Computer organisation** is the study of internal working, structuring and implementation of a computer system.
- It refers to the level of abstraction above the digital logic level, but below the operating system level.



5. Why Study Computer Organisation?

(From user to builder)

- You want to call yourself a **computer scientist/specialist**.
- You want to **build** software people use.
- You need to make purchasing **decisions**.
- You need to offer “expert” **advice**.
- Hardware and software affect performance
 - Algorithm determines number of source-level statements (eg: CS1010, CS2030, CS2040, CS3230)
 - Language, compiler, and architecture determine machine instructions (COD chapters 2 and 3)
 - Processor and memory determine how fast instructions are executed (COD chapters 5, 6 and 7)
- Understanding performance (COD chapter 4)

End of File