Welcome to The Hardware Lab!

Fall 2021
Logic Design Flow

Prof. Chun-Yi Lee

Department of Computer Science
National Tsing Hua University

Agenda

- Announcements
- Logic Design Flow
- Datapath and instructions

Today's class will help you:

1. Understand why we learn logic design
2. Understand the basic logic design flow
3. Understand the functions of different design phases

Announcements

- Lab 6
  - Lab 6 basic question demonstration on **12/2/2021 (Thu)**
  - Lab 6 advanced question demonstration on **12/16/2021 (Thu)**
  - Lab 6 car demonstration on **12/23/2021 (Thu)**
  - **Please read the lab description and specification carefully**
  - **Compile your codes again before merging yours with your teammates**
  - **Please follow the template I/Os and submit your .v files**

- Final project
  - You can begin working on your final projects from now
  - Please avoid copying the works from past years

Final Project

- Use your FPGA to implement creative and interesting works

- Final project proposal due on **12/13/2021 (Mon)**
- Final project report due on **1/14/2022 (Fri)**

Final Project Presentation

- Final project demonstration on **1/13/2022 (Thu)** and **1/14/2022 (Fri)**
  - Around 10 minutes per team

- **No restriction on your presentation style**
  - Be creative!
  - What is special and new in your project
  - Key features

- Order of presentation
  - We will randomly decide the order
  - Let us know if you have constraints

Final Project Report

- Final project report
  - Due on **1/14/2022, 23:59pm (Fri)**
  - Block diagrams and state transition diagrams
  - **Detailed explanation**

- Report contents
  - Introduction
  - Motivation
  - System specification
  - Experimental results
  - Conclusion
  - …And any other sections that you would like to include

Final Project Award

- **Category:** Difficulty and completeness
  - Graded by me and the TAs
    - Difficulty (**35%**)
    - Completeness (**30%**)
    - Source code coding style (**15%**) (Correctness, usage, comments, etc.)
    - Report (**20%**)
  - First place: **NTD $6,000**
  - Second place: **NTD $3,000**
  - Third place: **NTD $1,500**

- **Category:** Best creativity
  - Voted by everyone in the class
  - **NTD $1,000**

- **Category:** Deep learning models in FPGA
  - Extra **bonus points** and **cash rewards**. To be announced

Agenda

- Announcements
- Logic Design Flow
- Datapath and instructions

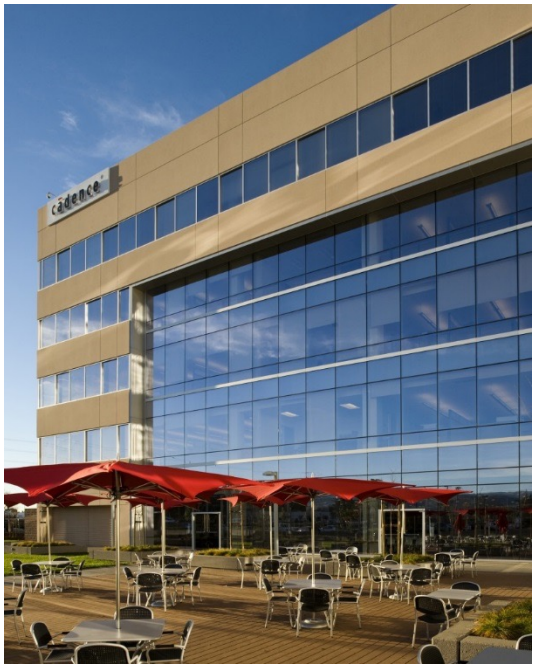
Today's class will help you:

1. Understand why we learn logic design
2. Understand the basic logic design flow
3. Understand the functions of different design phases

What is Computer-Aided Design

- Use **Software** to assist **Hardware** designs
- Also called **Electronic Design Automation (EDA)**
- Requires programming skills, data structures, and algorithms

Synopsys, Inc.



Cadence Design Systems, Inc.

*Pictures cited from www.synopsys.com and www.cadence.com for education purpose only

IC Design Flow

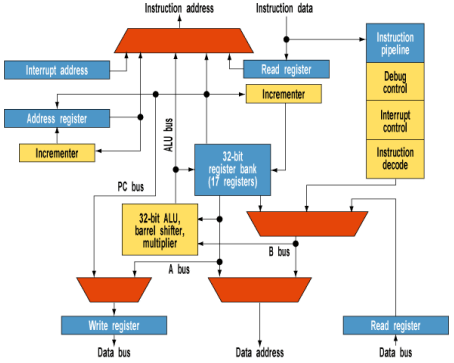
System specification

We are here

Verilog

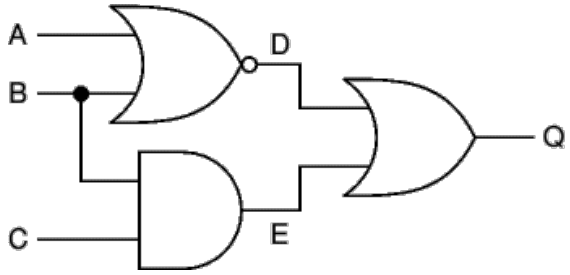
Logic and circuit design

Function and architecture design



The Cortex M3's Thumbnail architecture looks like a conventional Arm processor. The differences are found in the Harvard architecture and the instruction decode that handles only Thumb and Thumb 2 instructions.

Logic synthesis



*Pictures cited from maximintegrated.com and www.apple.com for education purpose only

IC Design Flow (Con'd)

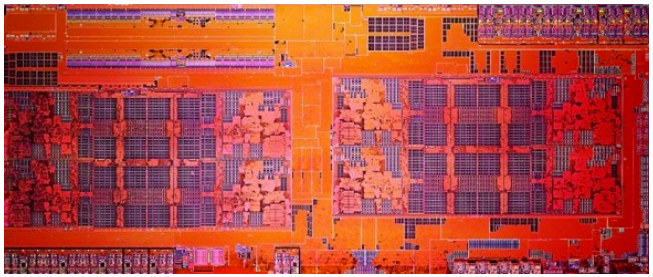
Verification

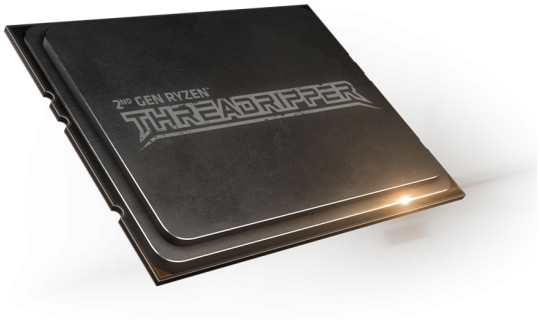
tsmc
台灣積體電路製造股份有限公司
Taiwan Semiconductor Manufacturing Company, Ltd.

Fabrication

Physical design

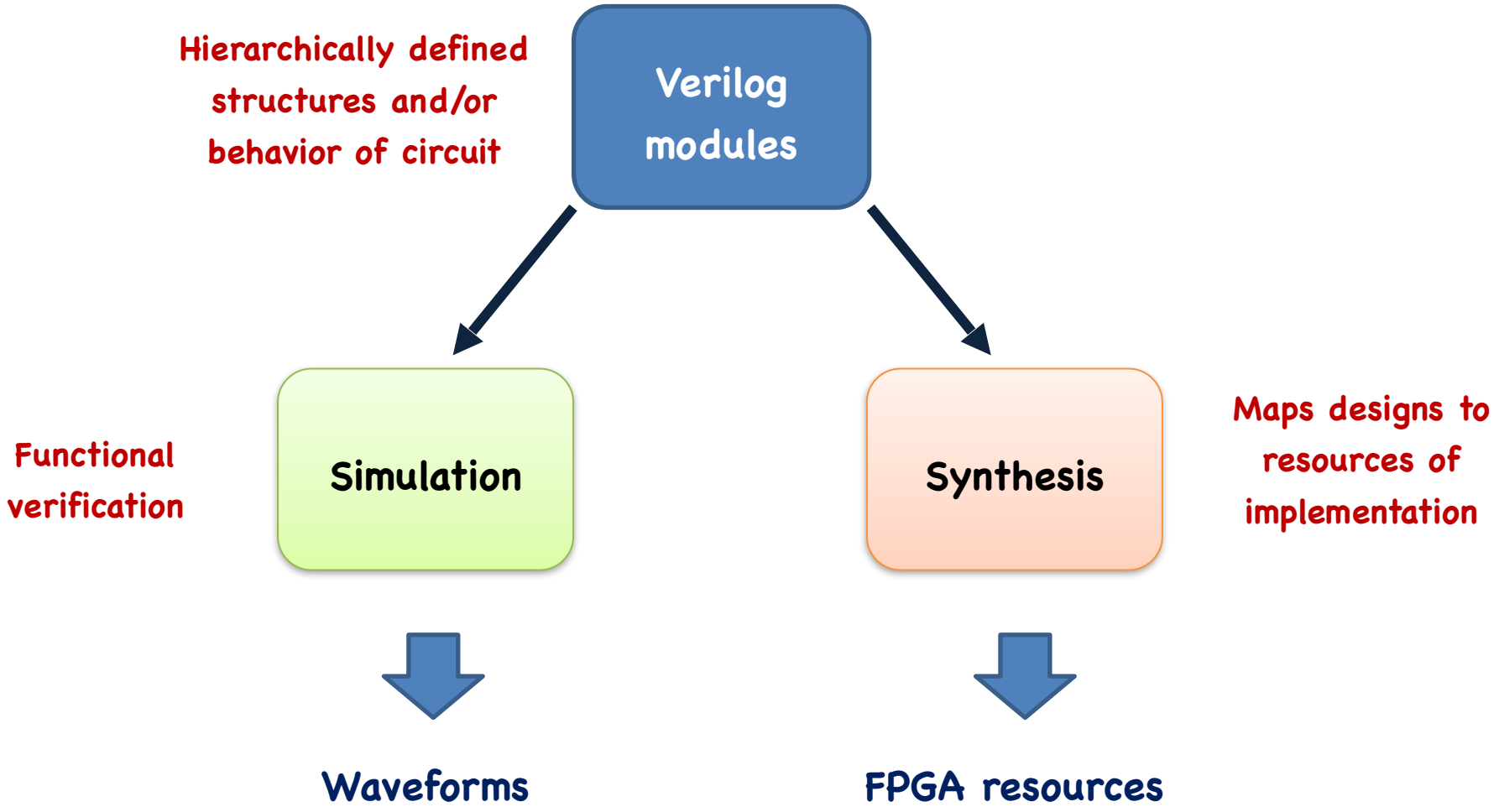
Packaging





*Pictures cited from maximintegrated.com and www.apple.com for education purpose only

Design Methodology in Hardware Lab



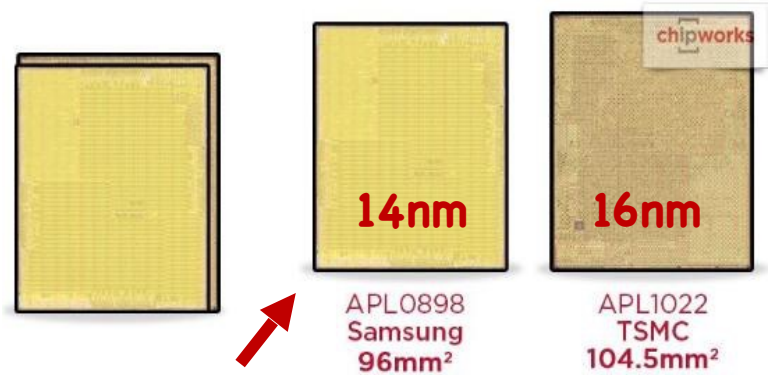
Logic Synthesis

- Verilog was originally a pure hardware simulation language
  - Similar to C, popular and widely used in the industry
  - People wrote programs to convert Verilog codes into low-level circuit descriptions [netlist]

- Netlists vary for different targets
  - For FPGAs: LUTs, flip-flops, and RAM blocks
  - For ICs: standard logic and memory libraries

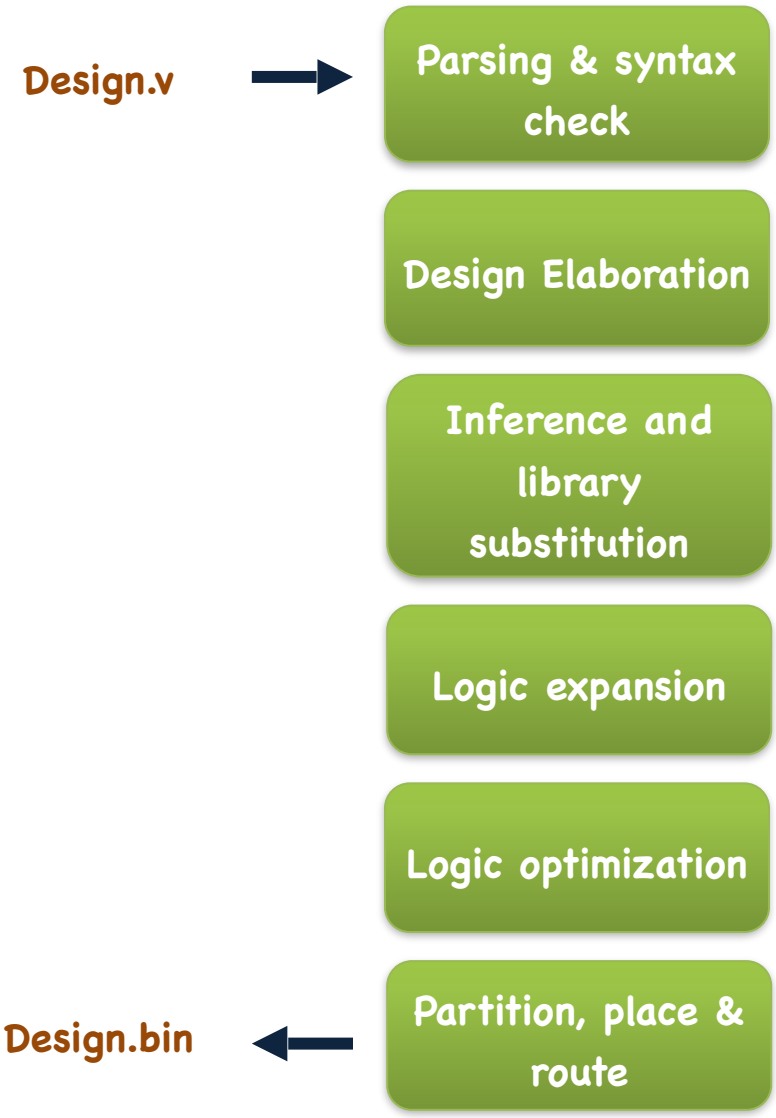
Why Logic Synthesis?

- Automatically manages details of the design process
  - Fewer bugs
  - Higher productivity
- Abstracts design from technologies
  - Designs can be re-synthesized targeting different chip technologies
  - E.g., Apple's A9 processor in TSMC's 16nm and Samsung's 14nm technologies
- Optimizes logic expressions



*Pictures cited from technews.tw for education purpose only

FPGA Logic Synthesis Steps

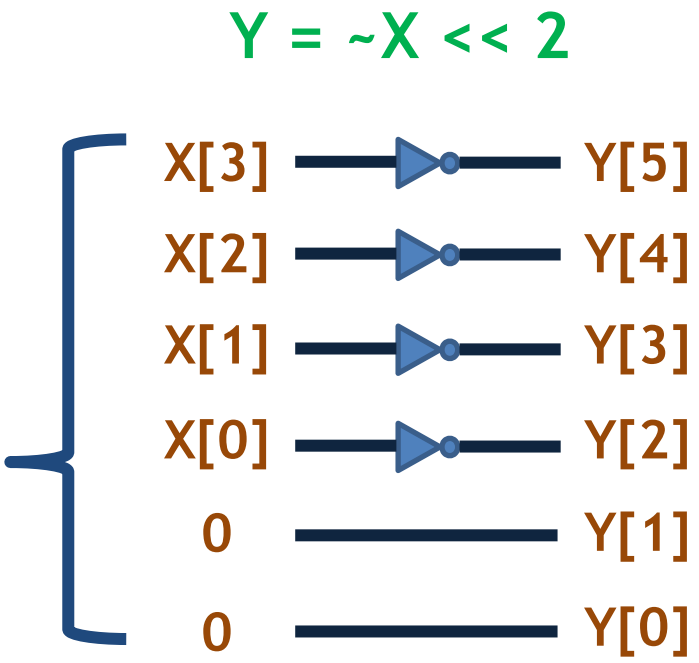


- Load in Verilog modules, run macro preprocessor for `define, `include, etc.
- Compute parameter expressions, create instances, connect ports
- Recognize and insert special blocks (memory, flip-flops, arithmetic blocks, etc.)
- Expand combinational circuits to primitive Boolean representation
- Apply Boolean algebra and heuristics to simplify and optimize under constraints
- Partition the design, map designs onto FPGA elements, assign physical locations, route connections

*Pictures revised from Prof. John Wawrzynek's Digital Design course slides from U.C. Berkeley for education purpose only

Operators and Synthesis

- **Logic operators** map into **primitive logic gates**
- **Arithmetic operators** map into **adders, substractors, multipliers, etc.**
- **Relational operators (>, <, =)** generate **comparators**
- **Shift by constant amount** are just **wire connections**
- **Shift by variable number of bits** generates a **shifter**
- **Conditional expressions** generates **muxes**

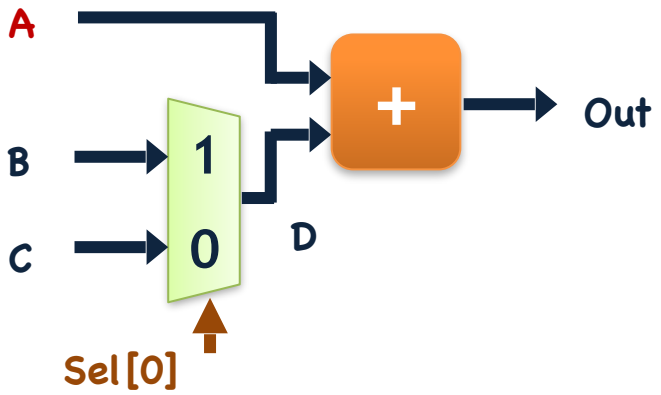


*Pictures revised from Prof. John Wawrzynek's Digital Design course slides from U.C. Berkeley for education purpose only

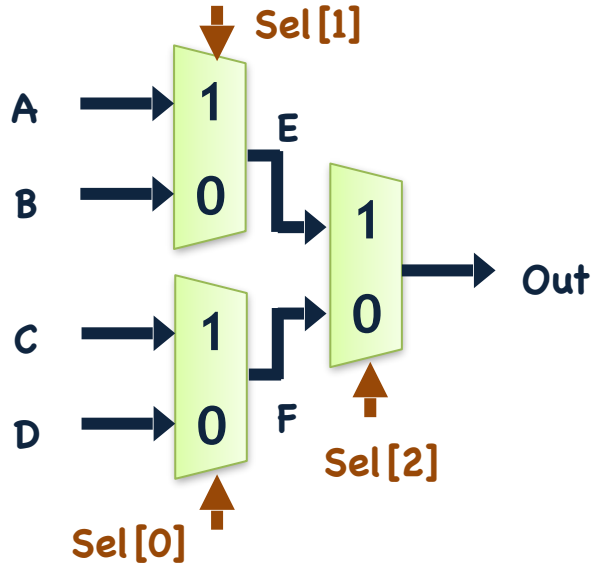
Conditional Operations

- **Conditional operations** are synthesized to **muxes**

```
always@(Sel or A or B or C or D)   begin
      Out = A + D;

      if (Sel[0] == 1'b1)      D = B;
      else                     D = C;

end
```




```
always@(Sel or A or B or C or D or E or F)
begin
      if (Sel[2] == 1'b1)  Out = E;
      else                 Out = F;

      if (Sel[1] == 1'b1)      E = A;
      else                     E = B;

      if (Sel[0] == 1'b1)      F = C;
      else                     F = D;
end
```



Synthesis Goal

- Use C/C++ programs to optimize your circuit
  - Simplify your logic circuits
  - Target: area, timing, and power
  - **Requirements**: algorithms, data structures, etc.
- Technology-independent
  - Optimization of logic expressions
- Technology dependent
  - Technology mapping (which technology to use)
  - Cell library binding (which logic gate library to use)

Famous Logic Synthesis Tools

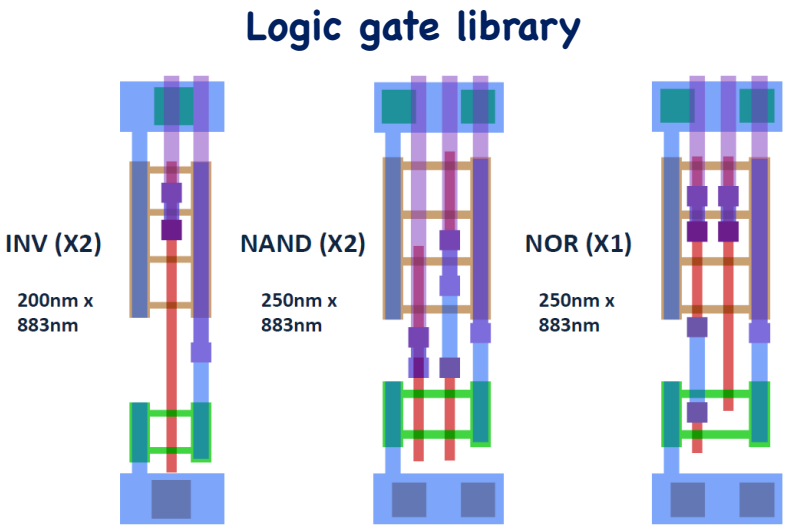
- **ASIC (Application-specific integrated circuit)** synthesis tools
  - **Design Compiler** and **IC compiler** by **Synopsys**
  - **Encounter RTL Compiler** by **Cadence Design Systems**
  - **HDL Compiler** by **Mentor Graphics**

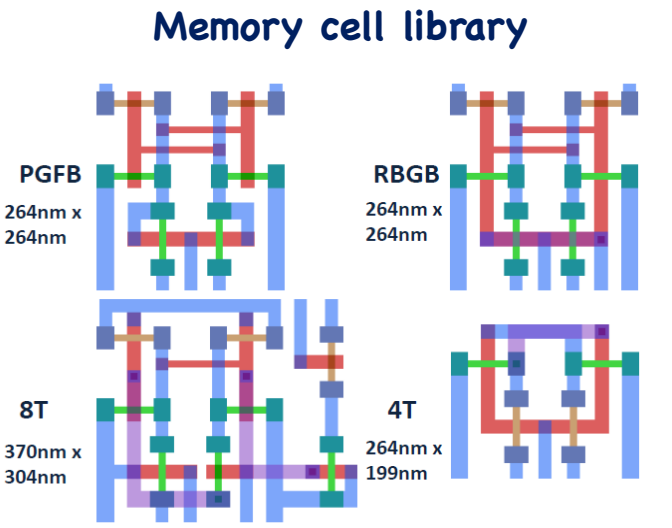
- **FPGA** synthesis tools
  - **ISE** and **Vivado** by **Xilinx**
  - **Quartus II** by **Altera (acquired by Intel)**
  - **Encounter RTL Compiler** by **Cadence Design Systems**

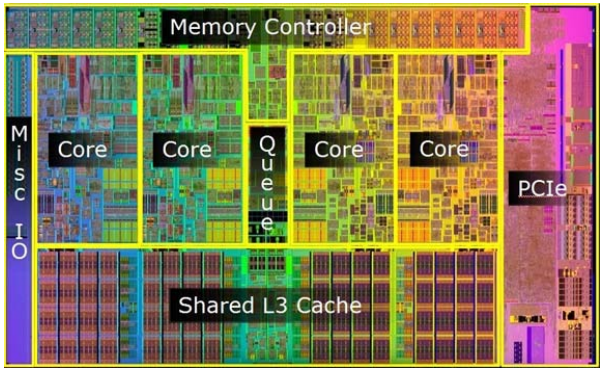
*Pictures cited from www.wikipedia.com for education purpose only

Standard Cell Library

- Provided by IC fabrication companies
  - E.g., TSMC, UMC, etc.



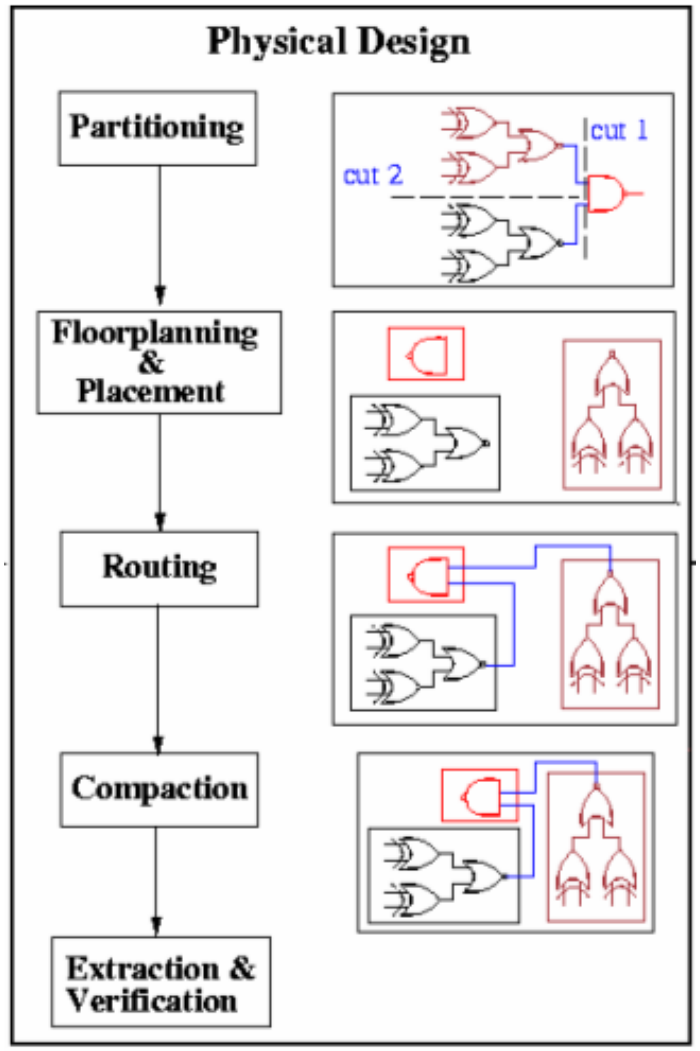


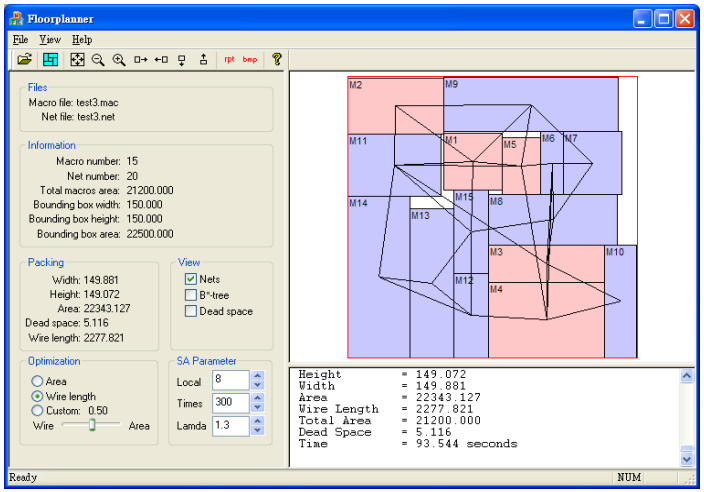


Intel I7 core

*Pictures cited from www.intel.com for education purpose only

Physical Design





Floorplanning system

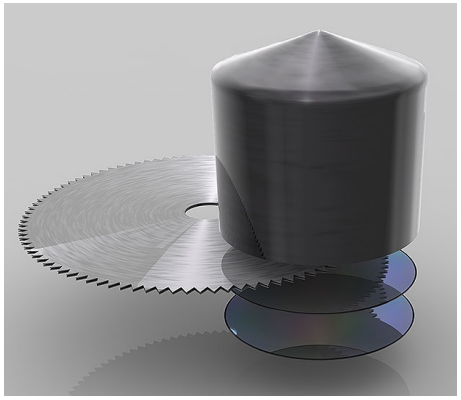
Routing system

*Pictures cited from Prof. Yao-Wen Chang's slides from National Taiwan University for education purpose only

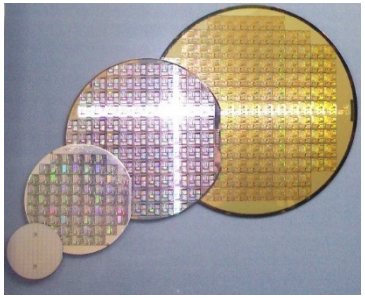
Physical Design Verification

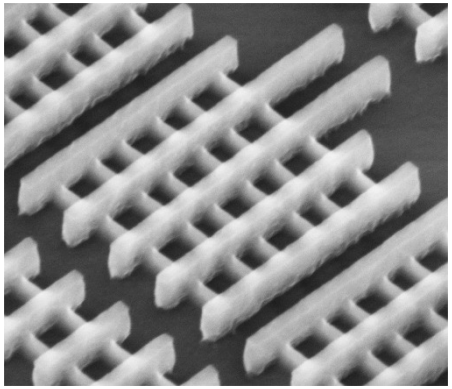
- Physical design rules define by TSMC or UMC
  - Check distances among standard cells and transistor components
  - Width and distance of metal wires, transistor components, etc.
- Varies from technology to technology
- Rules also differ from company to company
- Checks correctness of the final layout design
  - Design rule checking (DRC)
  - Layout versus schematic (LVS)
  - Electrical rule check (ERC)

TSMC Chip Fabrication

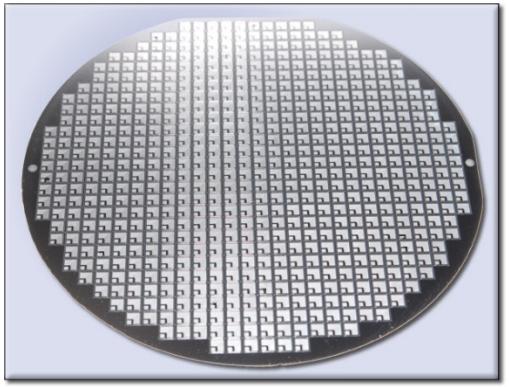


Ingot fabrication and cutting into wafers





Transistor fabrication



Cut wafers into dies (bare chips)

十萬青年十萬肝　台積電夜鷹計畫救台灣

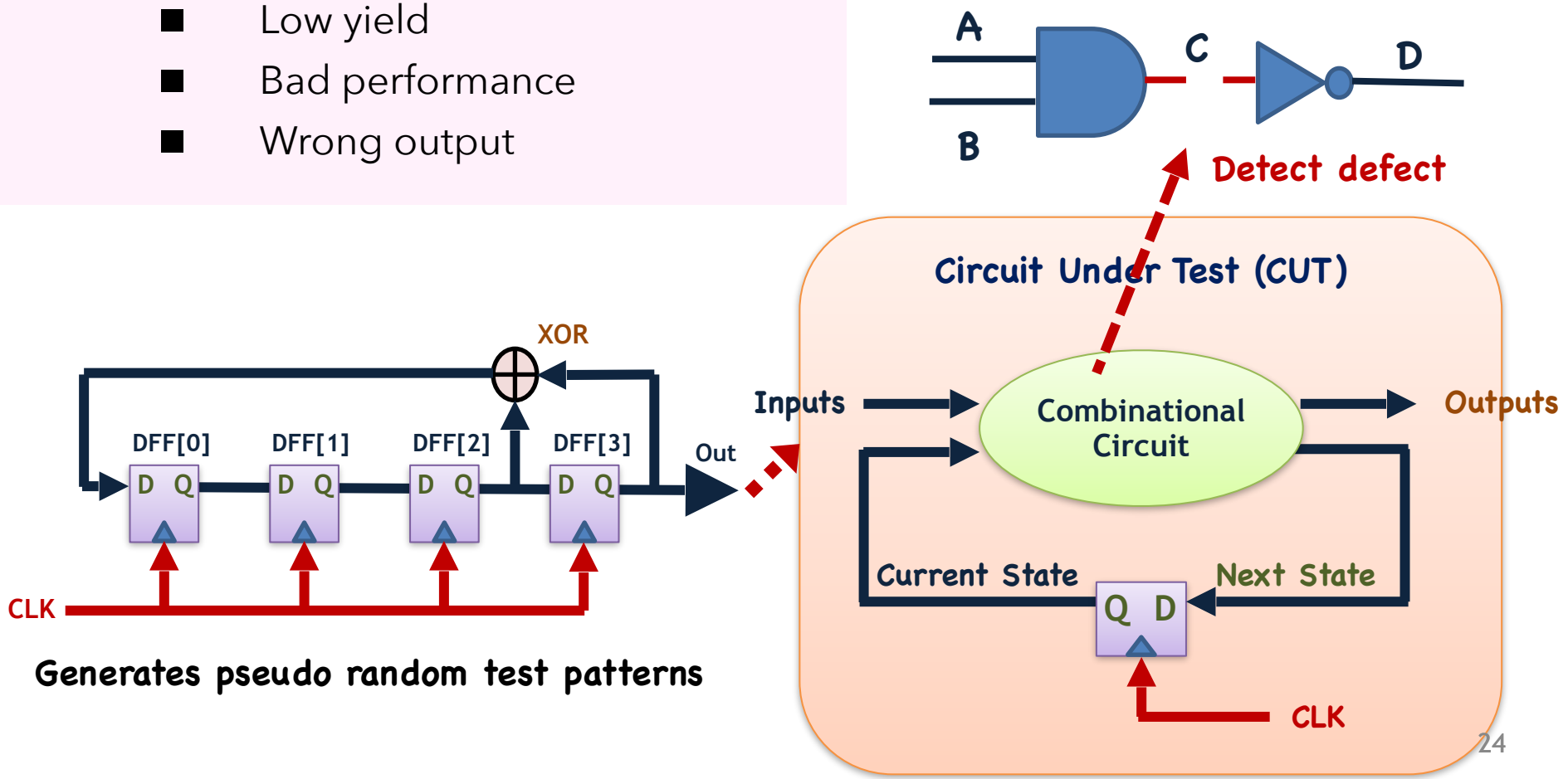
徐嶽煌／北美智權報 總編輯　2014.06.17

晶圓代工大廠台積電為了在十奈米製程領先群雄，特別在內部進行夜鷹計畫，預計招募一批十奈米技術開發夜鷹部隊，來投入大夜班的研發輪值，希望做到24小時研發不間斷，以達到今年股東會年報上所說的2016年量產目標。並以底薪加30%、分紅加50%的方式，希望重賞之下必有勇夫，如同當年國軍「**十萬青年十萬軍**」的氣魄般，募集人力、讓肝功能指數發揮最大效應、一口氣達陣。

*Pictures cited from apcmag.com, www.wikipedia.com, www.transyselectronics.com, and www.naipo.com for education purpose only

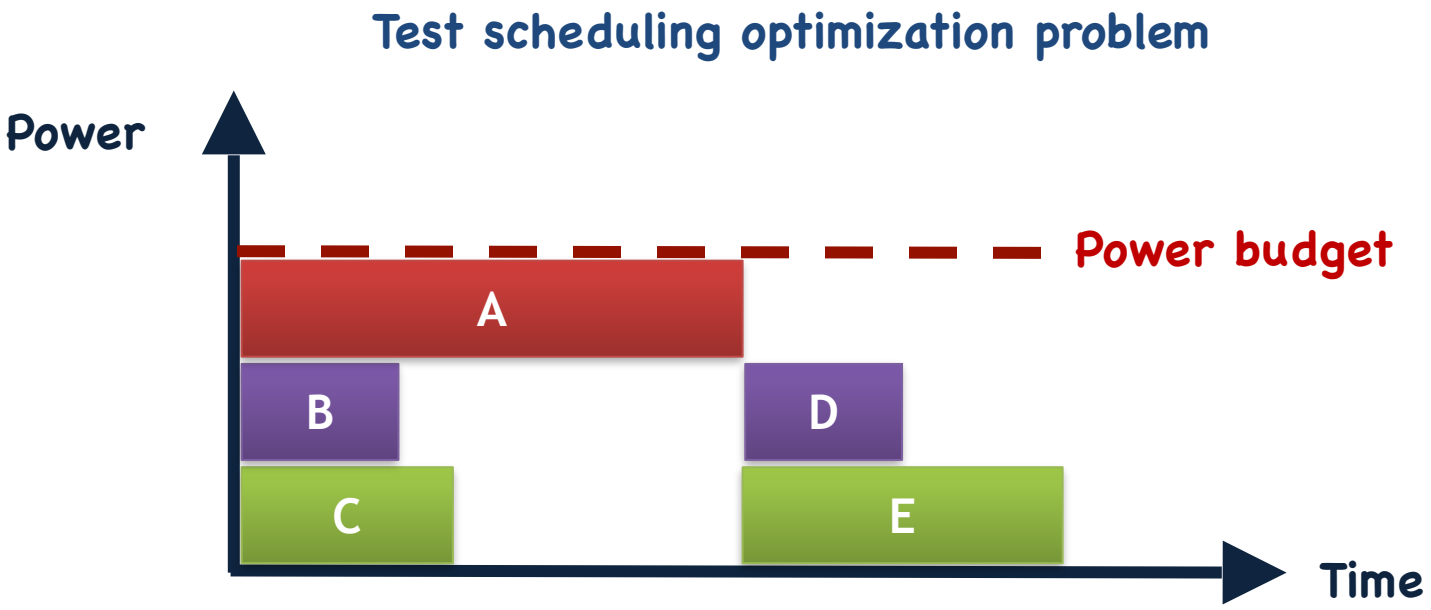
VLSI Testing

- To ensure the quality of chips
- Fabrication may cause errors
  - Low yield
  - Bad performance
  - Wrong output



VLSI Testing

- VLSI Testing is also an area which requires strong programming skills
  - Build-in self test
  - Test pattern generation
  - Low power test
  - Test scheduling



Agenda

- Announcements
- Logic Design Flow
- **Datapath and instructions**

Today's class will help you:

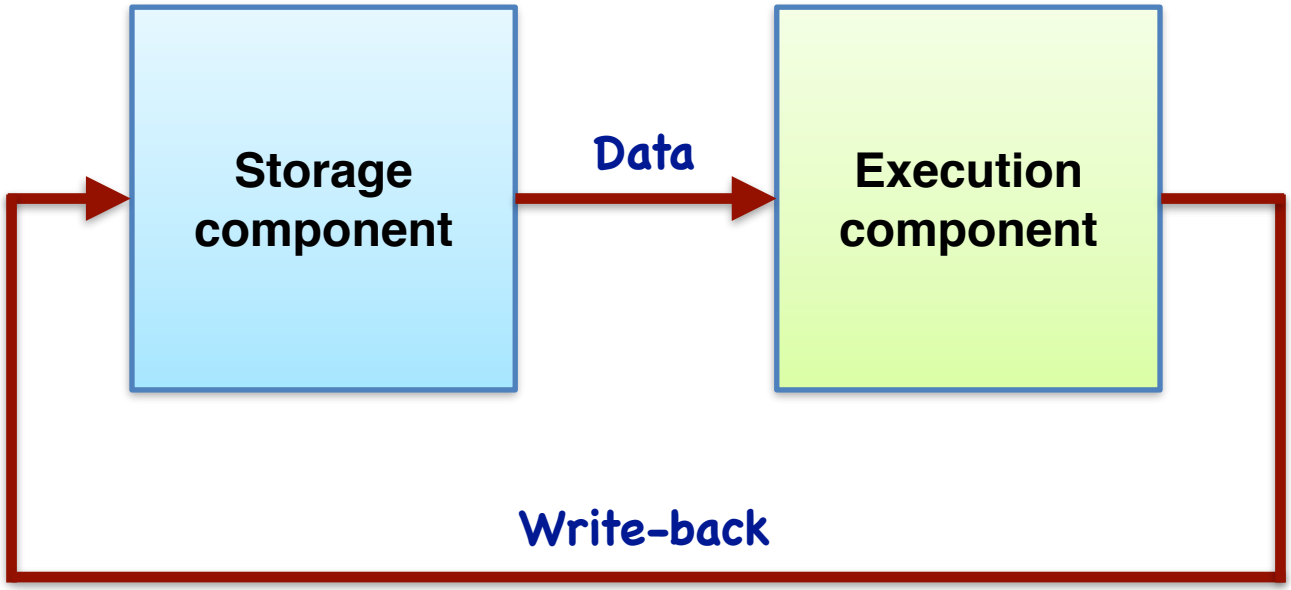
1. Understand why we learn logic design
2. Understand the basic logic design flow
3. Understand the functions of different design phases

Datapath Components

- Design a system that contains control, execution, and storage components
  - **You have already learned all of these components in this semester**
- Data are stored in **storage components**
  - Memory
  - Cache
  - Registers
- Data are processed in **execution components**
  - Arithmetic components (adders, multipliers, subtractors, etc.)
  - Logical operation components (AND, OR, NOT, etc.)
  - Shifters and comparators
- Data flow is determined by **control components**

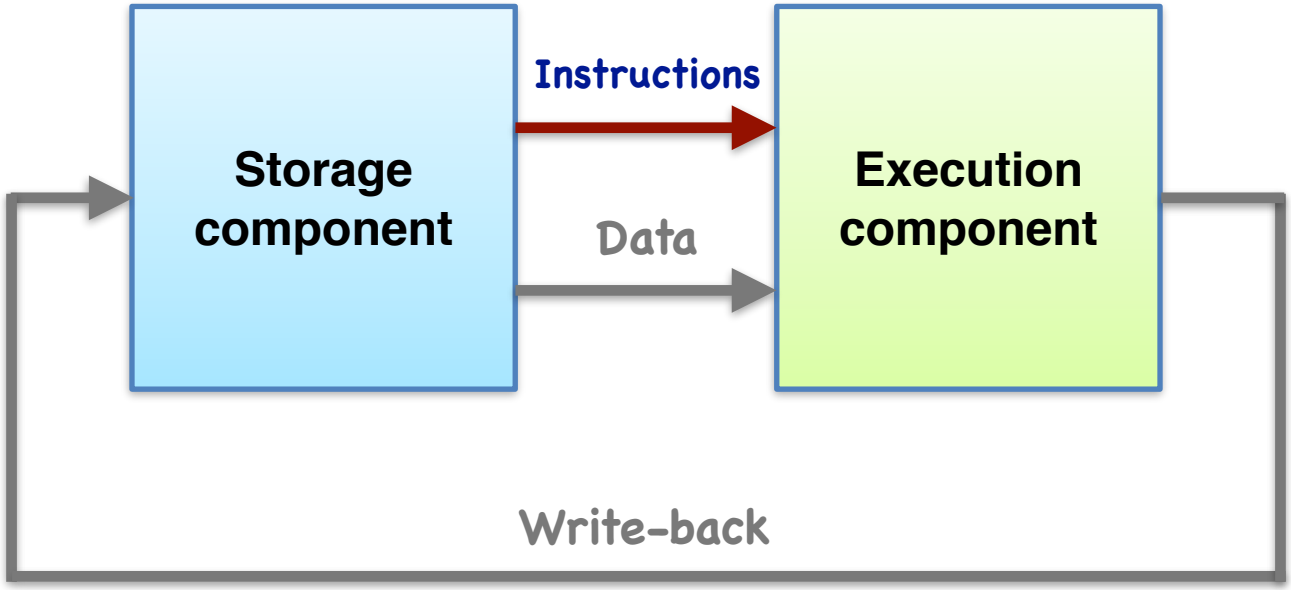
Simple Datapath Structure

- Storage components provides data to be processed
- Execution component process the data
- The result is stored back to the storage component



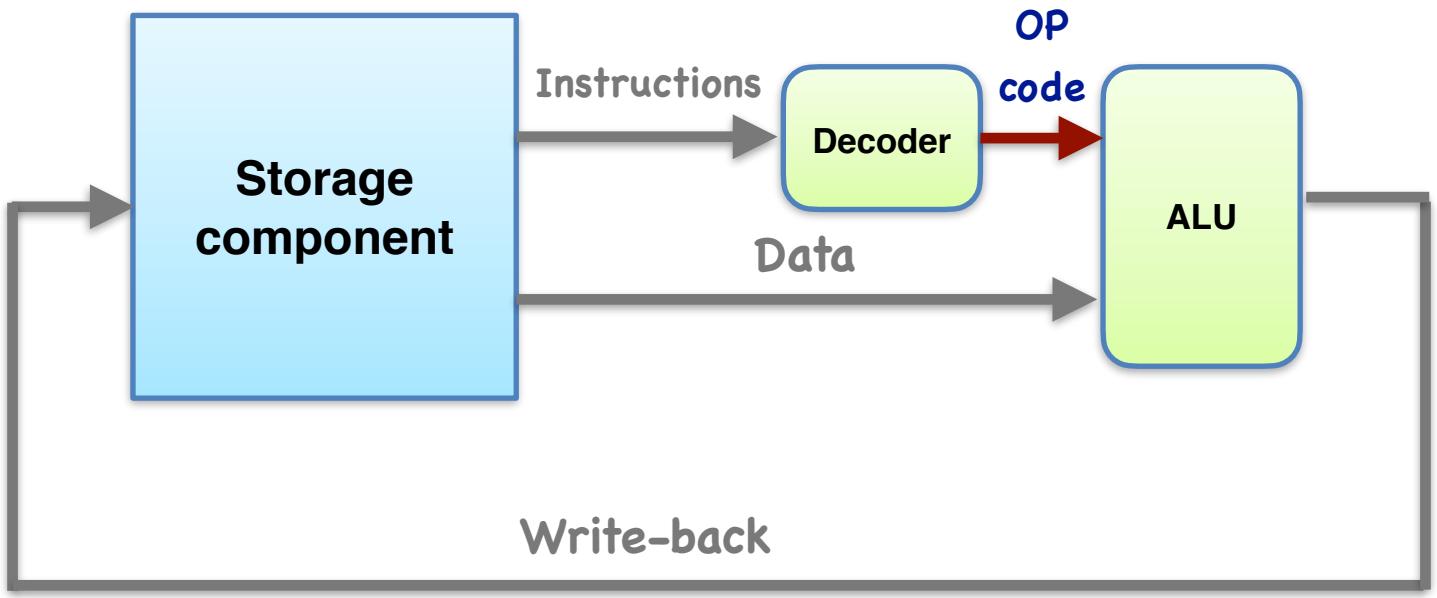
Instructions Comes In

- Instructions tells the execution component what operations to execute
- Depending on the instructions, the execution component selects the appropriate sub-units to run



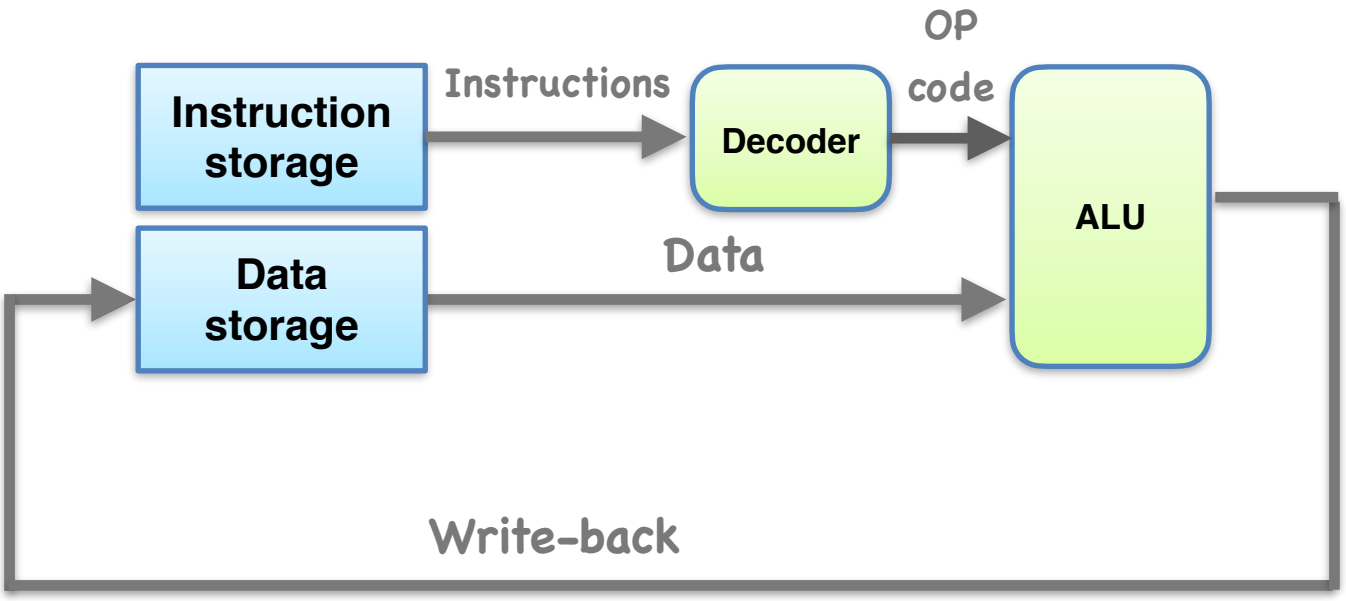
Decoder Comes In

- ALU stands for **A**rithmetic **L**ogic **U**nit
- Instructions are decoded by an decoder (Lab 1)
- The decoder generates an OP code for ALU (Lab 2)
- The ALU contains adders, multipliers, shifters, etc for execution (Lab 1 & 2)



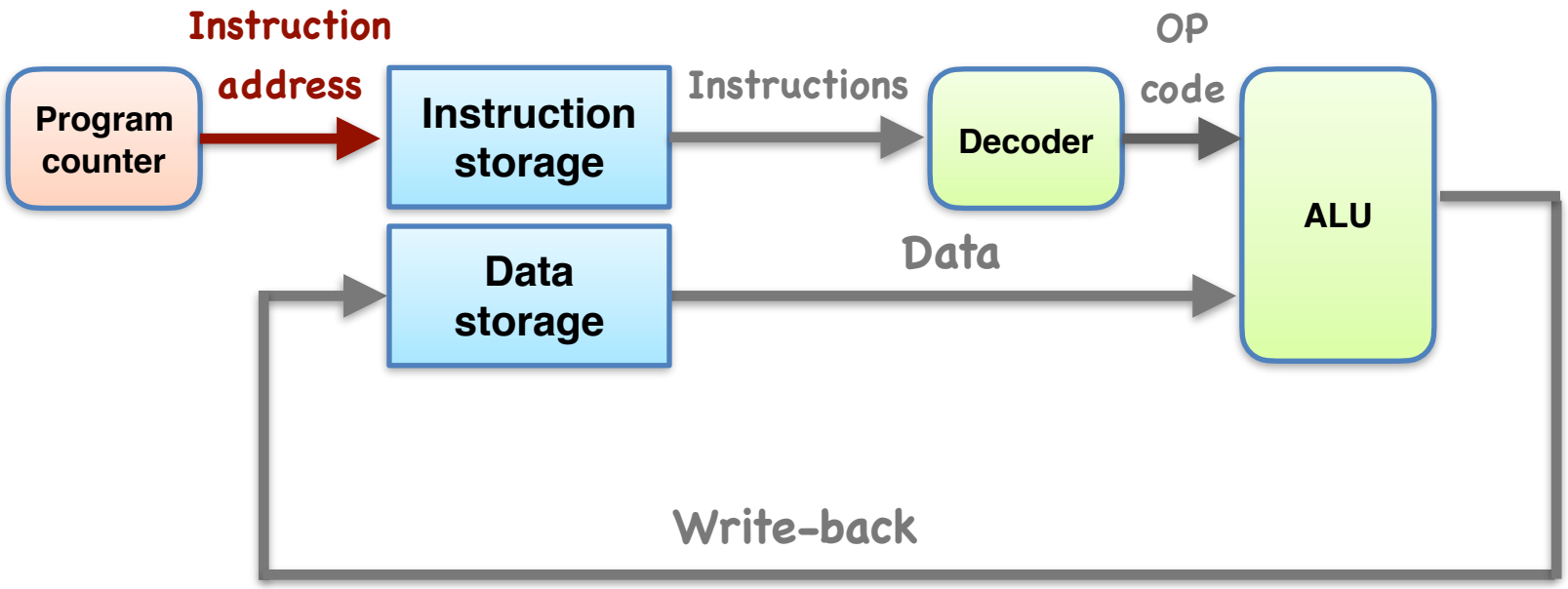
Separation of Storage Component

- The storage component can be splitter into instruction storage and data storage
- The instruction storage stores the instructions to be operated
- The data storage stores the data to be processed



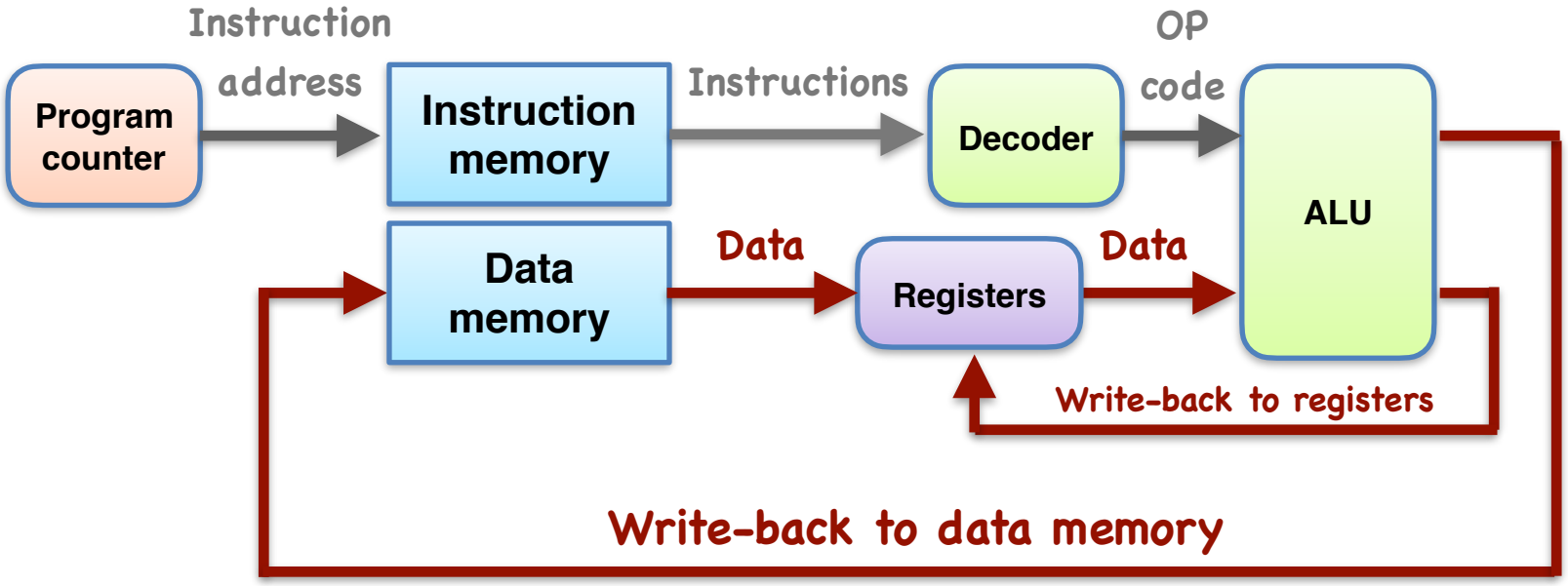
Program Counter

- Program counter (PC) provides the address of the instruction to be executed in the instruction storage
- Program counter updates the address to the next instruction every cycle
  - PC <= PC + 4 (4 bytes per instruction)
- The instructions stored in the instruction storage are called **a program**



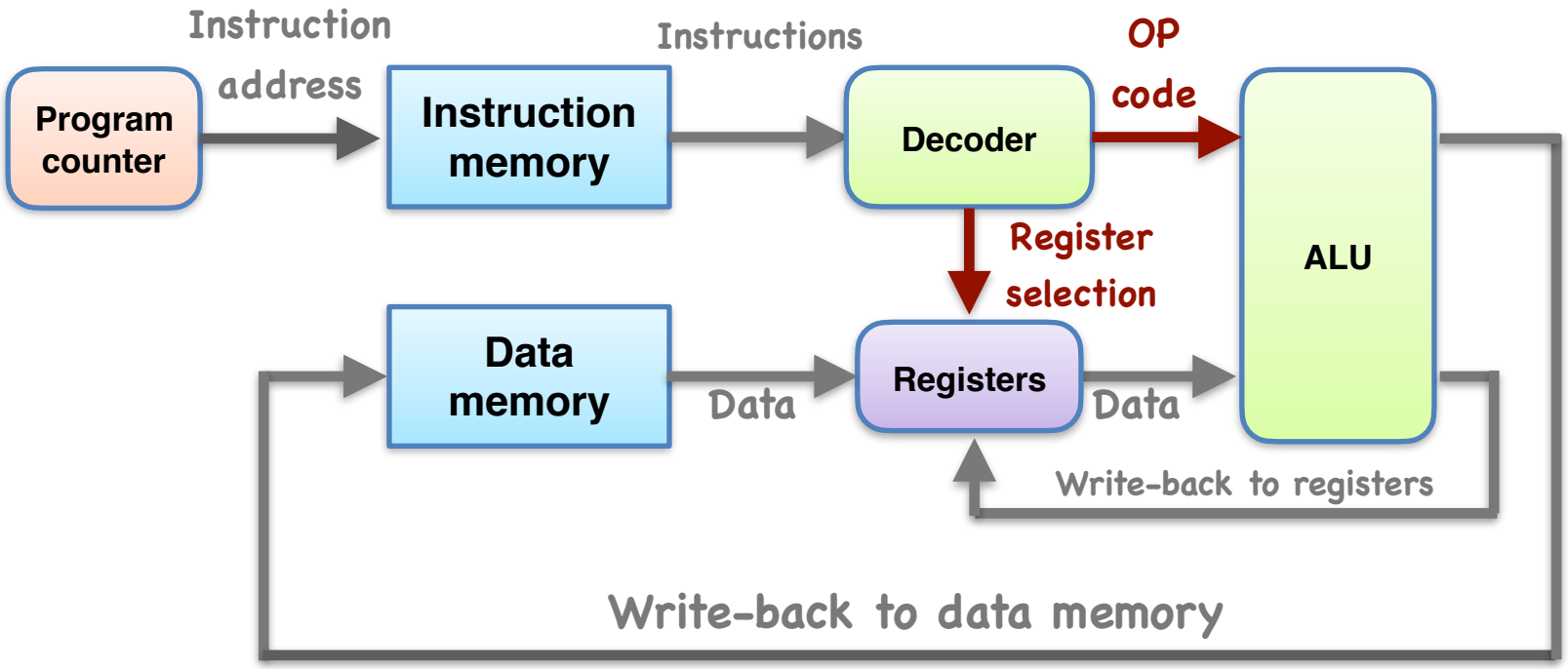
Registers for Local Storage

- Data storage is separated into registers and data memory
- Registers are small amount of storage elements constructed from DFFs
- Data can be loaded from the data memory to the registers
- Data can also be stored back to the registers from the ALU results



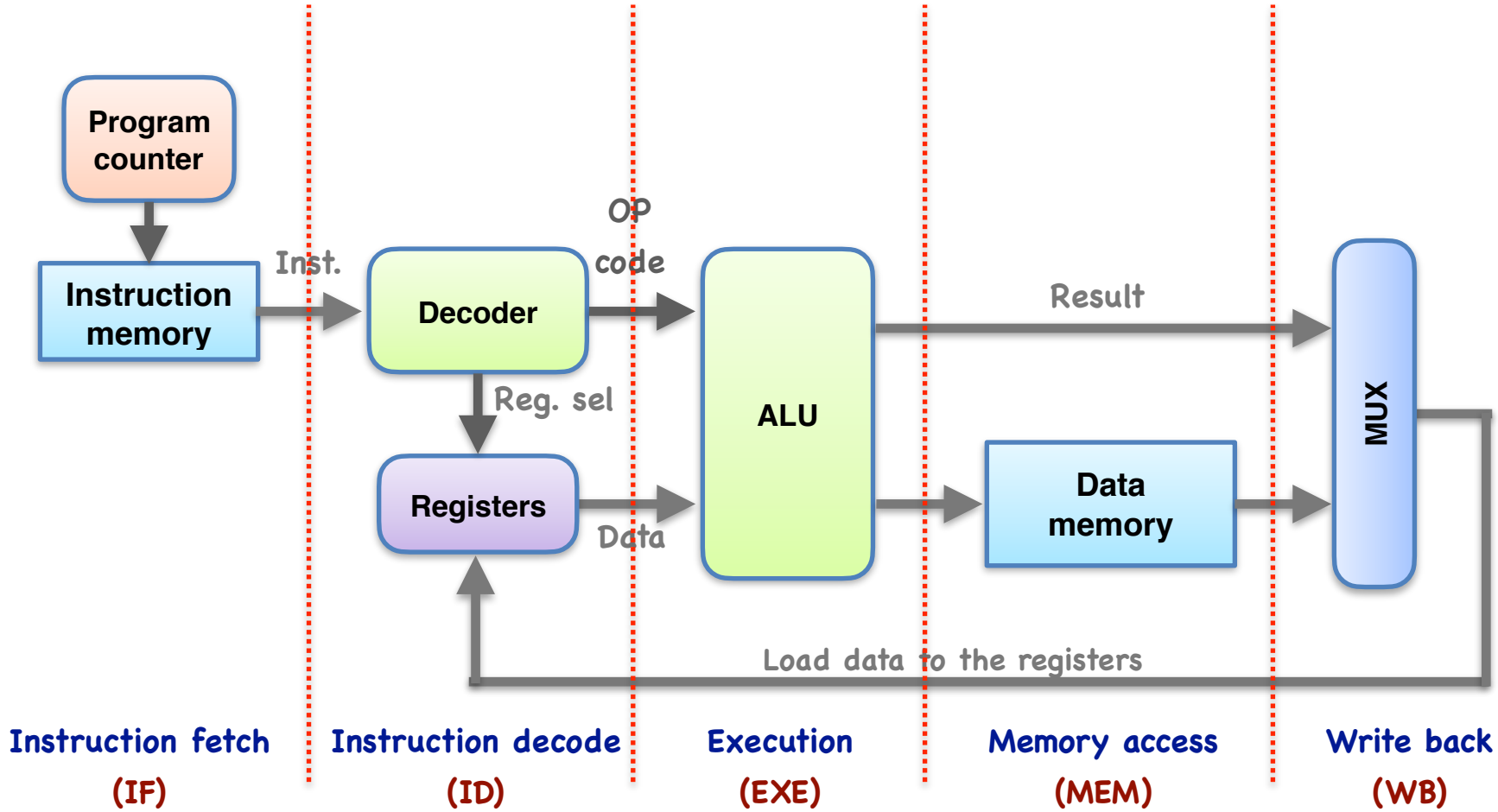
Selection of Registers

- The decoder generates two items: **OP code** and **register selection**
- The decoder selects the registers whose contents are to be fed into ALU



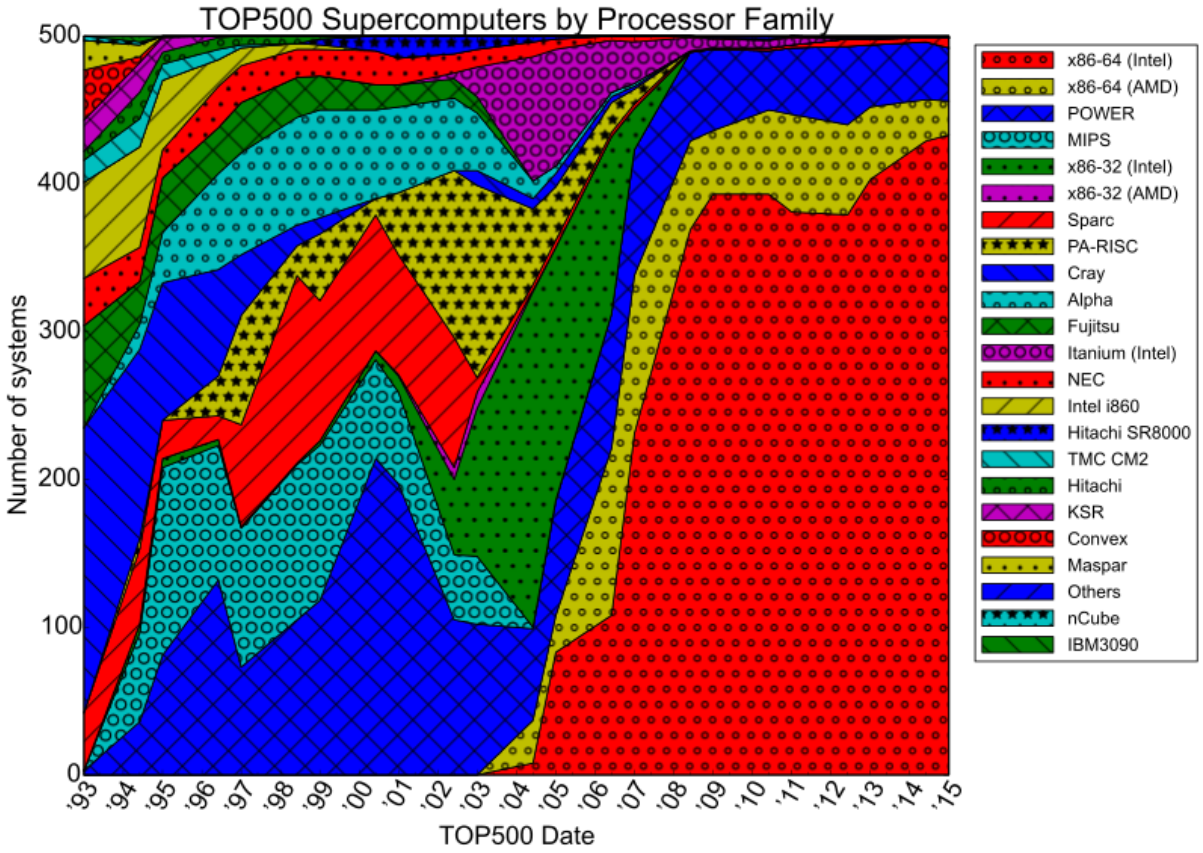
Five Stage Pipeline

- The datapath can be re-organized and separated into five stages
- This five stage pipeline structure is the basis of computer architecture



What is Instruction Set?

- A set of instructions supported by the underlining hardware
- Different vendors provide different ISAs
- Abstracts hardware for programmers



*Pictures cited from en.wikipedia.org/wiki/X86-64 for education purpose only

The MIPS Instruction Set

- Stanford MIPS commercialized by MIPS Technologies (www.mips.com)
  - MIPS I, MIPS II, MIPS III, MIPS IV, MIPS V, **MIPS32**, and **MIPS64**
- Large share of embedded core market
  - **Embedded systems**: Windows CE devices, routers, gateways, **Amazon Kindle**
  - **Game consoles**: Nintendo 64, Sony PlayStation 1, 2, and Portable
  - Applications in consumer electronics, network/storage equipment, cameras, printers, etc.
- Typical of many modern ISAs

How Does an Instruction Look Like

- Fixed instruction formats
- The syntax of the instructions depends on their "types"

Arithmetic operations

Logical operations

Memory operations

Conditional operations

Miscellaneous Instructions

Arithmetic Operations

- The simplest type of MIPS instructions
- Three **operands**
  - **Operands** are the variables to be operated
  - Two sources and one destination
  - **Syntax**: <**Opcode**> <**Dest**>, <**Src1**>, <**Src2**>

- Take a piece of C code for example: **a = b + c**

MIPS Instruction

add a, b, c

It means a gets the value of (b + c)

- **Philosophy: Keeping the hardware as simple as possible**

Simplicity Favors Regularity

- Take another piece of C code for example: **a** = **b** + **c** + **d**

MIPS Instruction

```
add  a, b, c
add  a, a, d
```

It means a gets the value of (b + c) first, then add d to itself

- **Why don't we have the add instruction take one more operand?**
  - **Regularity** makes implementation simpler
  - **Simplicity** enables higher performance at lower cost

Format of Register Operands

- Arithmetic instructions use register operands
  - Each operand represents a **register**

- MIPS supports **32 registers**, each of them contains **32 bits**
  - Use for frequently accessed data only (recall the library example)
  - 32-bit data is called a "**word**"
  - **Philosophy**: **Smaller is faster**

Registers located inside a processor
(1) Fast (2) Expensive (3) Limited

- Types of registers
  - **$t0, $t1, …, $t9 for temporary values**
  - **$s0, $s1, …, $s7 for saved variables**
  - **$v0, $v1 for results and expressions**
  - **$a0, $a1, …, $a3 for arguments (procedures and functions)**

Register Operand Example

- Let's replace the operands by registers
- Take another piece of **C code** for example

$$f = (g + h) - (i + j)$$

- Map of values in the registers

f	g	h	i	j
$s0	$s1	$s2	$s3	$s4

MIPS Instruction

```
add  $t0, $s1, $s2
add  $t1, $s3, $s4
sub  $s0, $t0, $t1
```

Immediate

- In MIPS assembly, **immediate** means a **constant**
- Many instructions allows **immediate** to be used as operands

MIPS Instruction

```
addi  $t0, $t1, 42
```

Adds $t1 with 42 and stores the result in $t0

MIPS Instruction

```
li  $t0, 42
```

Loads 42 into $t0

- **Immediate may also be expressed in hexadecimal: 0xFFFFFFFF**

Logical Instructions

- Logical instructions also have **3 operands**
  - Two sources and one destination
  - **Syntax: <Opcode> <Dest>, <Src1>, <Src2>**

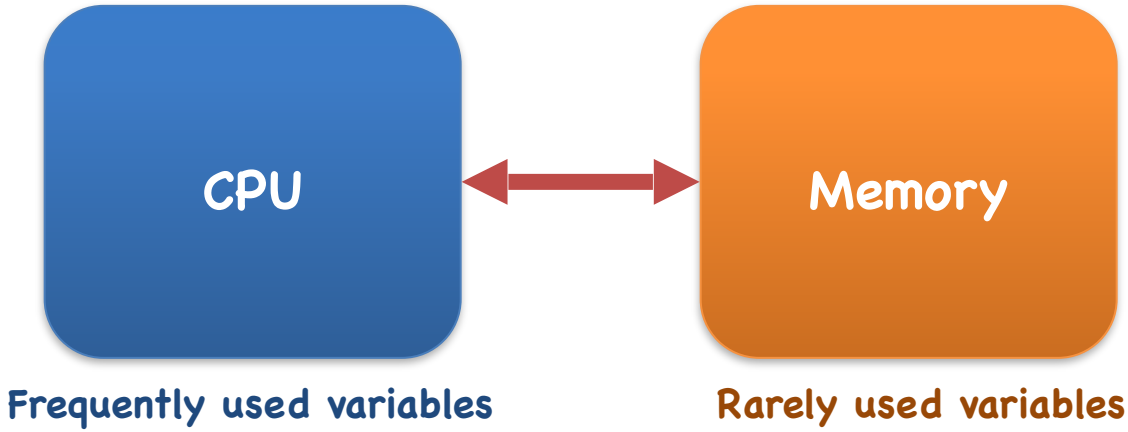
```
and  $s0, $s1, $s2      # bitwise AND
andi $s0, $s1, 42
or   $s0, $s1, $s2      # bitwise OR
ori  $s0, $s1, 4
nor  $s0, $s1, $s2      # bitwise NOR2
sll  $s0, $s1, 10       # logical shift left
srl  $s0, $s1, 10       # logical shift right
```

Note: NOT is not supported by MIPS: Can be done by NOR

Registers vs. Memory

- Operand to arithmetic & logical instructions must be **register** or **immediate**
  - Registers are limited (**32 registers** in MIPS)

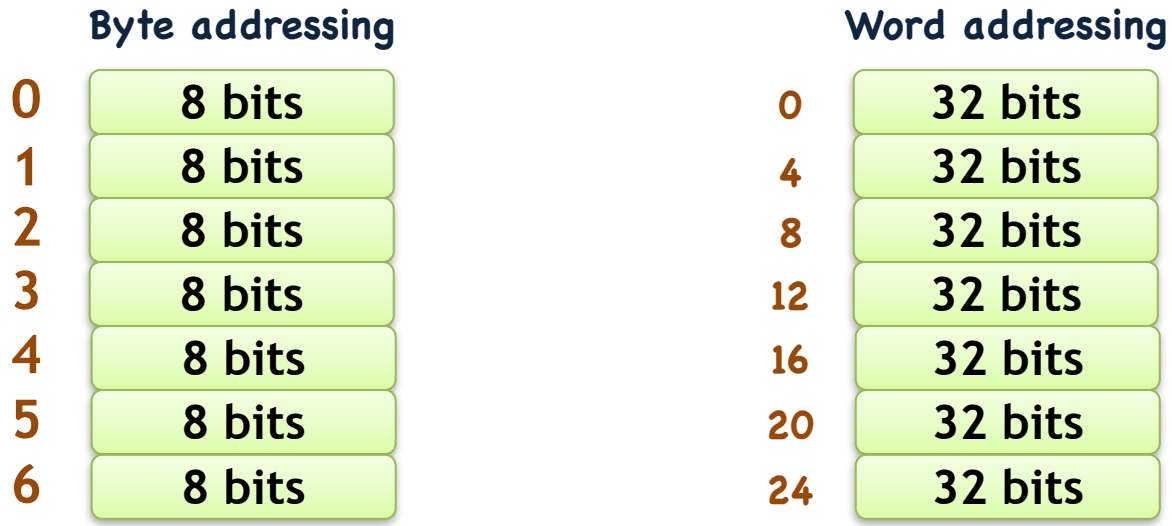
- When writing a program, it may contain arbitrary number of variables
  - Variables have to be mapped to registers before operations
  - **Register allocation**
  - What about programs with lots of variables?



Memory Organization

- Memory is a large, single-dimensional array, with an address
  - An address serves as an index to the memory array
  - Byte addressing

- Abstract view of a memory array
  - A word is 32 bits or 4 bytes (register size)



Memory Operations

- Transfer data between memory and registers
- Load and store are the most basic operations
- Three **operands**
  - Two registers (one stores the address), and one immediate (as offset)
  - **Syntax**: <**Opcode**> <**Src/Dest**>, **immediate** (<**base**>)
- Take a piece of C code for example: **a[12] = b + a[8]**

MIPS Instruction

```
lw  $t0, 32 ($s3)      // Loads a[8] into $t0
add $t0, $s2, $t0      // Add b and a[8]
sw $t0, 48 ($s3)       // Stores $t0 into a[12]
```

Assuming that $s3 points to the address of a[0]

Thank you for your attention!

*Pier view taken at Sausalito, California, USA.
This picture is taken by Chun-Yi Lee himself, who is also a fan of photography