CO 2206 Introduction to Computer Organization

Ong Wee Hong

weehong.ong@ubd.edu.bn, owh@ieee.org, G38 IPI SHOAS

Universiti Brunei Darussalam

Acknowledgement

The content of the slides used in this course are extracted from various sources including those quoted under references in following slides, teaching material from UBD lecturers who had taught this course before and from the material received from the author's course of studying

- Jan 2009

References

- Books
  - Computer Organization and Architecture: Designing for Performance, 7th edition by William Stallings
  - Logic and Computer Design Fundamentals, 4th edition by Mano and Kime
- Web pages
  - William Stallings resources
    - http://williamstallings.com/COA/COA7e.html
  - Mano and Kime resources
    - http://www.writphotec.com/mano4/index.html
  - Introduction to RISC Technology
    - http://www.inf.fh-dortmund.de/personen/professoren/swik/risc/intro_to_risc/irt0_index.html

Course Content

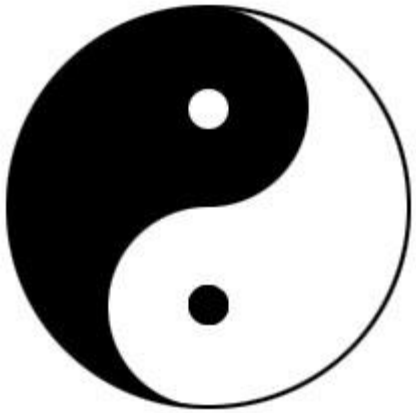
- Introduction
- Building Blocks of Computer
  - Digital Logic
  - Logic Circuit Design
  - Registers and Register Transfer
- Computer Design Basics
- RISC and CISC
- Memory Basics
- IO and Buses

Course Management

- Learning activities (subject to alternative arrangements)
  - Lectures
    - 8:00-9:50, FSM 2.18, Monday
  - Laboratories and tutorials
    - 8:00-9:50, FSM 1.19, Thursday
- Assessment scheme
  - Examination 70% - 180 min exam in May
  - Coursework 30% - few problem solving

CO2103 + CO2206

- The two courses are tightly related and complement each other
- Together, they cover the important linkage and interfacing between hardware and software
- In computing world, hardware and software must co-exist
  - Software without the hardware to execute is useless
  - Software gives intelligence to the hardware



Why take these courses? - 1

Why bother about hardware and its details?

- You want to be a plastic surgeon
- You all know how to use knife, needle, thread
- To be successful you don't just "cut & paste"
- You have to learn about the body
  - Skin
  - Bones
  - Muscles
- Different parts of the body require different skills
  - Nose
  - Hands
  - Legs
- Who do you trust performing the surgery?

Why take these courses? - 2

Why bother about hardware and its details?

- You want to be a race car driver
- You all know how to drive
- To be successful you don't just drive
- You must "be in touch with your vehicle"
- You have to learn about the vehicle
  - Engine
  - Suspension
  - Tires
- Is it drag racing, monster trucks, NASCAR, endurance
  - Different cars
  - Different style of driving
- Who is going to win the race?

Why take these courses? - 3

Why bother about hardware and its details?

- You want to be a Computer Scientist
- You all know how to program
- To be successful you don't just program
- You have to understand the machine
  - Hardware: Processor, memory, disk, etc.
  - SW: Operating system, Programming Languages/Compilers
- What kind of computer scientist?
  - Databases, networks
  - Scientific computing (motion of planetary bodies, drug evelopment, computational biology, economics, etc.)
  - Games, virtual reality
  - Embedded: Cell phones, mp3 player, cars
- Who's code do you want controlling your brakes, airbag, financial transactions? Script kiddie or computer scientist.

Which code is “better”?

- it depends on what do you mean by “better”

```
main:
     ;multiply n1 by 10
     xor ax,ax
     mov cx,10
sum: add ax,[n1]
     loop sum
     ...
```

```
main:
     ;multiply n1 by 10
     xor ax,ax
     mov bx,[n1]
     mov cx,10
sum: add ax,bx
     loop sum
     ...
```

Which code is “better”?

- ;tranversing a 5 elements array

```
main:
     ;repeated codes
     xor ax,ax
     xor di,di
     add ax,[n+di]
     add di,2
     add ax,[n+di]
     add di,2
     add ax,[n+di]
     add di,2
     add ax,[n+di]
     add di,2
     add ax,[n+di]
     ...
```

```
main:
     ;using loop
     xor ax,ax
     xor di,di
     mov ax,[n]
     mov cx,4
sum: add di,2
     add ax,[n+di]
     loop sum
     ...
```

```
main:
     ;using subroutine
     xor ax,ax
     xor di,di
     mov ax,[n]
     mov cx,4
sum: call sub
     loop sum
     ...

sub: add di,2
     add ax,[n+di]
     ret
```

Architecture vs Organization - 1

- Synonymous in many uses and textbooks
- Architecture is those attributes visible to the programmer
  - Instruction set, number of bits used for data representation, I/O mechanisms, addressing techniques.
  - e.g. Is there a multiply instruction?
- Organization is how features are implemented
  - Control signals, interfaces, memory technology.
  - e.g. Is there a hardware multiply unit or is it done by repeated addition?

Architecture vs Organization - 2

- Computer Architecture can have a number of organizational implementations
  - control signals
  - technologies
  - device implementations
- Computer Organization is transparent to the programmer, however it is the main contributing factor in improving computer performance (pipelining, branch prediction, latency hiding, etc)

Architecture vs Organization - 3

A computer's architecture is its abstract model and is the programmer's view in terms of instructions, addressing modes and registers. A computer's organization expresses the realization of the architecture. Architecture describes what the computer does and organization describes how it does it.

Architecture and organization are independent; you can change the organization of a computer without changing its architecture. For example, a 64-bit architecture can be internally organized as a true 64-bit machine or as a 16-bit machine that uses four cycles to handle 64-bit values.

Architecture vs Organization - 4

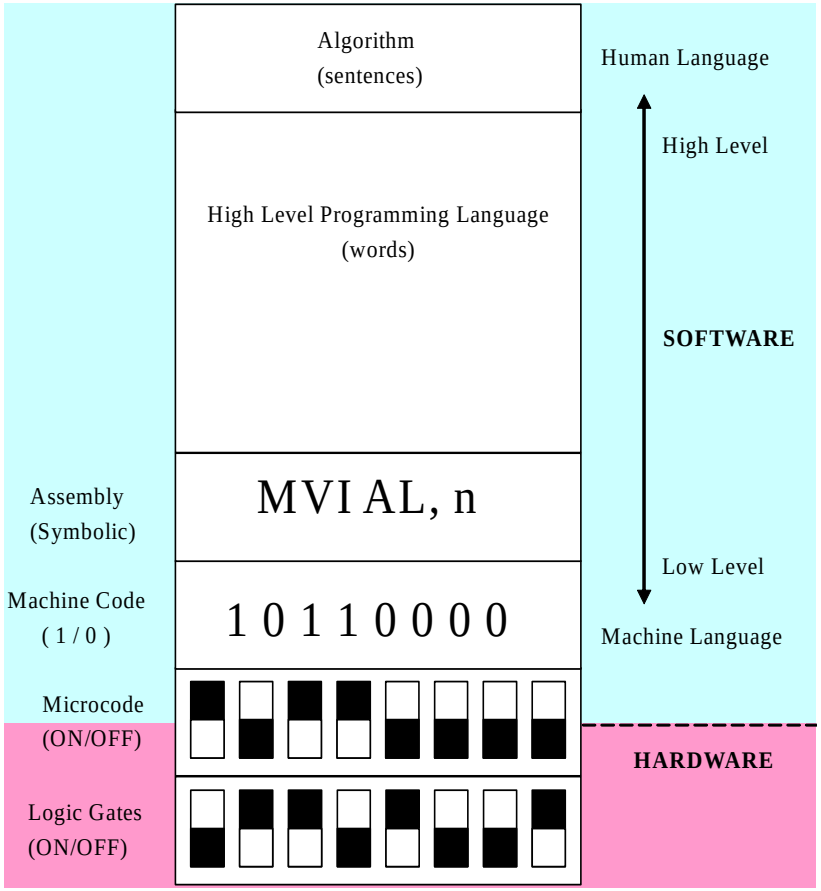
- All Intel x86 family share the same basic architecture
- The IBM System/370 family share the same basic architecture
- This gives code compatibility
  - At least backwards
- Organization differs between different versions

Architecture vs Organization - 5

- It may not be important to distinguish between the attributes of architecture and organization
- They are interrelated and sometimes difficult to distinguish
- Let's learn them together

Levels of Abstraction - 1

- Generalize a computer into different levels
- From the programmer's point of view:
  - High-Level Programming Languages
  - Assembly Language
  - Machine Code
  - Microcode (CISC)
  - Logic Gates

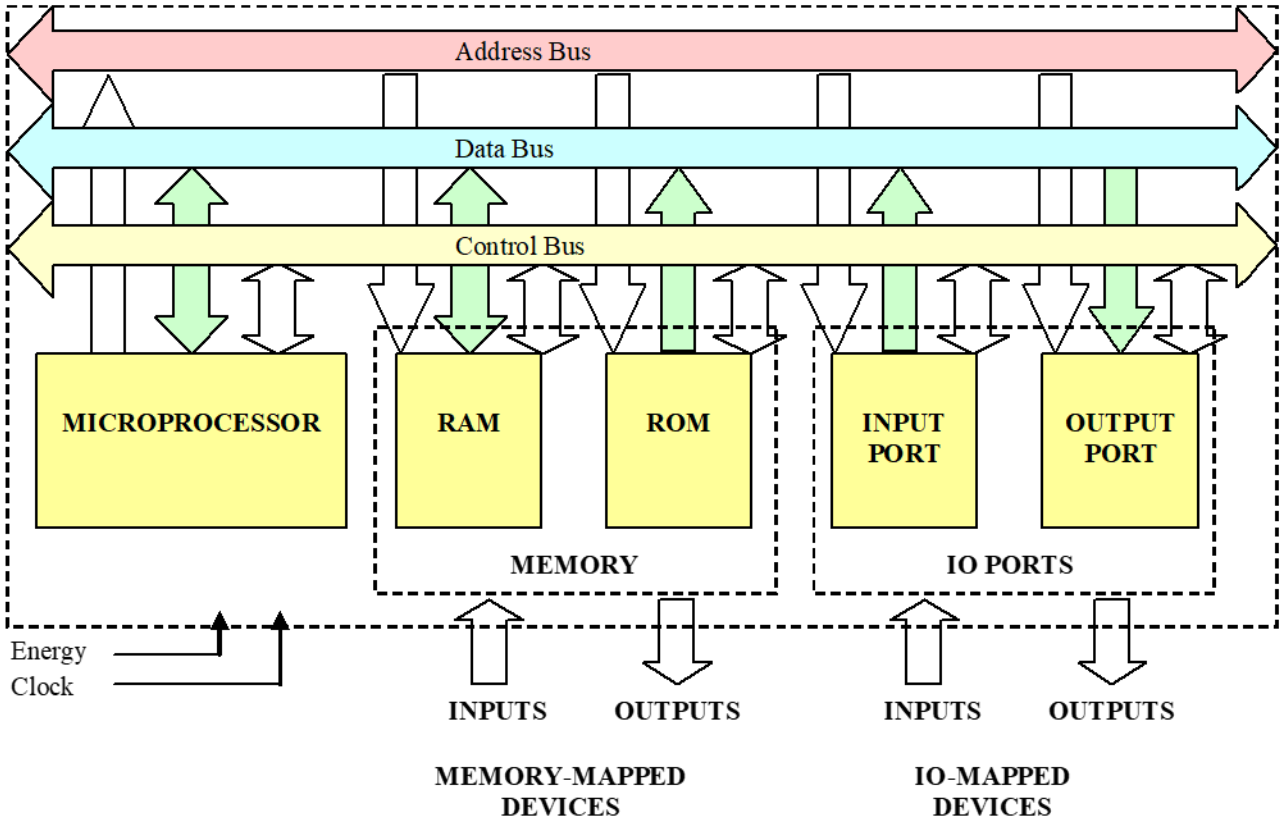


Levels of Abstraction - 2

- From the User's Point-of-View:
  - Applications Software
    - Word Processor, Spreadsheet, etc.
  - Operating System Software
    - UNIX, MS-DOS, OS/2, VMS, etc.
  - Hardware
    - Mainframe, Workstation, Personal Computer
- Applications are written for a specific Operating System
  - Operating System shields the Application from the Hardware
  - Different combinations of Hardware Platform, Operating System and Applications are possible

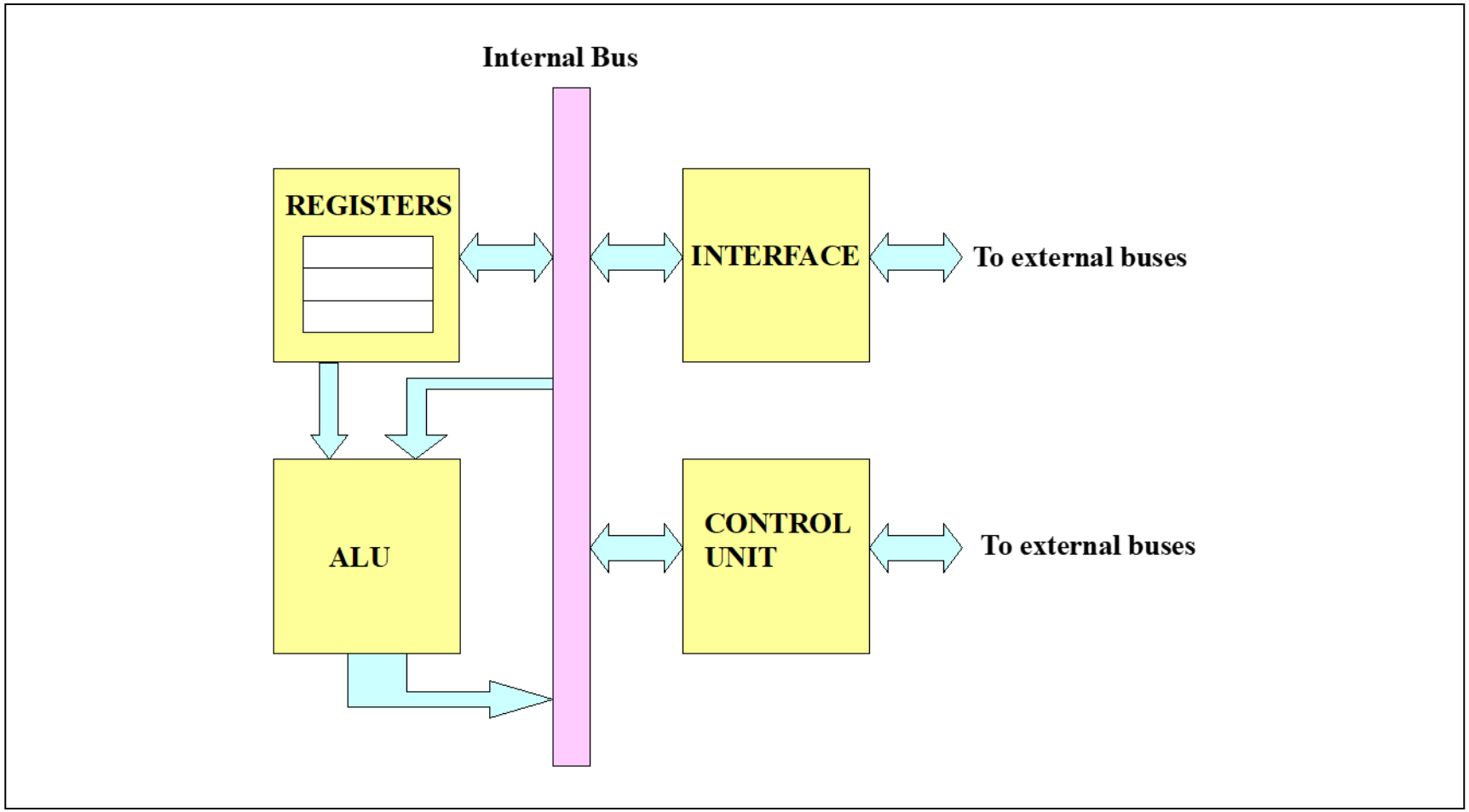
Not forgetting the big picture

General microcomputer organization



Microprocessor (uP) Organization

Just one of the models



Computer or uP Organization

- Majority of the topics will be on the organization of the microprocessor
- What makes up the different components in the microcomputer and microprocessor?
- What different (hardware) implementations or technologies are used to improve the performance?

Design Strategies/Approach
for digital circuits

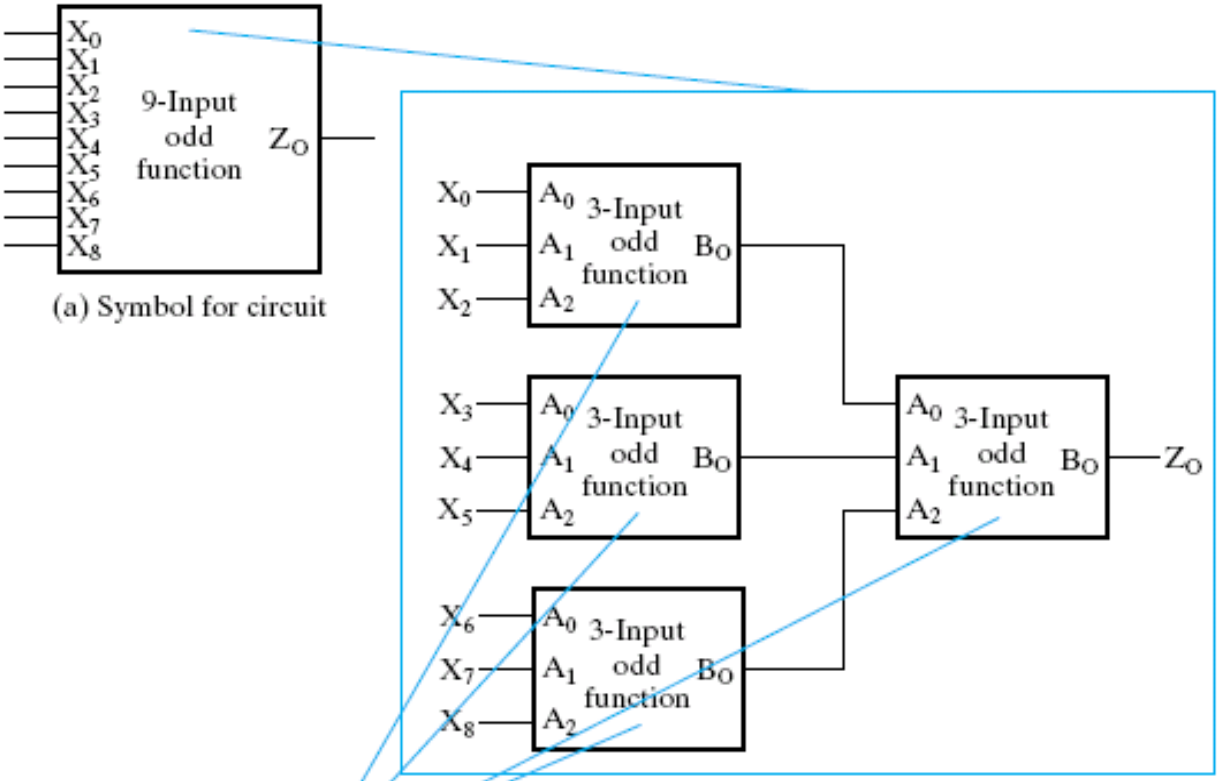
- Two general approach
  - *Bottom-up* (old style)
  - *Top-down* (mainstream)
- ***Bottom-up***
  - connect up gates, circuits to build up bigger systems, e.g. computer system
  - "we have these, so we can do these"
- ***Top-down***
  - mainstream design approach
  - focus on function, i.e. "we want to do these, so we need/make these"
  - repeatedly breaking down into compositional subsystems or blocks until base elements
    - bottom-up to build up the system

Top-down Design

- Made possible with sufficient bottom-up knowledge and functional blocks, i.e. “we have enough knowledge and experience to build the necessary building blocks”
- Circuit is specified by text or hardware description languages (HDL)
  - plus constraints on cost, performance and reliability
- In automated synthesis, HDL description is converted to logic automatically
- The logic is optimized and then mapped to available *primitive elements*
  - *primitive elements* are devices at lowest level of the design, e.g. *logic gates* for *logic* design at *gate* level
  - may be at *functional block* level
- In order to maximise reusability and satisfy constraints, it is often necessary to perform parts of the design bottom-up

Top-down: Hierarchical Design

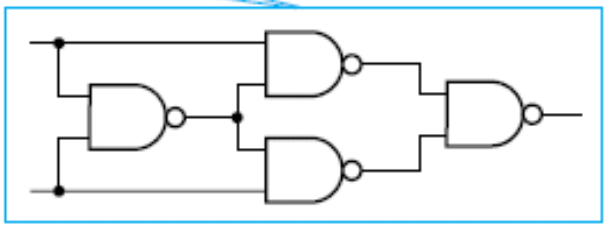
- ***Divide and conquer*** approach
  - Circuit is broken up into blocks
  - Blocks are interconnected to form the circuit
  - Blocks can be broken down into smaller, more manageable blocks
- A *hierarchical design*
  - Reduces the complexity of the schematic (logic diagram) and gives a simplified representation of a complex circuit
  - The function of a block can be defined by a program or description, instead of logic schematic
  - Blocks are reusable in the same circuit and possible in other design as well, thus reduces the design effort
    - Reusable blocks are *functional blocks*
    - CAD tool libraries may contain these predefined functional blocks



(a) Symbol for circuit

(b) Circuit as interconnected 3-input odd function blocks

(c) 3-input odd function circuit as interconnected exclusive-OR blocks



(d) Exclusive-OR block as interconnected NANDs

Summary

- Important for computer scientists to know hardware
- Computer organization and architecture cover the hardware aspect and its interfacing to software
  - for performance improvement
- Architecture describes what the computer does and organization describes how it does it (implementation)
- Topics learned in CO2103 will be required