CENG3420 Computer Organization & Design

Ryan Chan

March 6, 2025

Abstract

This is a note for **CENG3420 Computer Organization & Design** for self-revision purpose ONLY.
Some contents are taken from lecture notes and reference book.
Mistakes might be found. So please feel free to point out any mistakes.
Contents are adapted from the lecture notes of CENG3420, prepared by Bei Yu, as well as some online resources.

Contents

Chapter 1

Introduction

This course is about how computers work.

1.1 The Manufacturing Process of Integrated Circuit

For this chapter, only a few calculations need to be considered:

1. Yield = The proportion of working dies per wafer.
2. Cost per die $= \dfrac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{Yield}}$
3. Dies per wafer $\approx \dfrac{\text{Wafer area}}{\text{Die area}}$ (since wafers are circle)
4. Yield $= \dfrac{1}{\left[1 + \left(\frac{\text{Defects per area} \times \text{Die area}}{2}\right)\right]^2}$

> **Remark.** Note that the defects on average = Defects per unit area × Die area.

1.2 Power

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

> **Example.** For a simple processor, the capacitive load is reduced by 15%, voltage is reduced by 15%, and the frequency remains the same. Then, how much power consumption can be reduced?
>
> **Solution:**
>
> $$1 - (1 - 15\%) \times (1 - 15\%) \times 1 = 27.75\%$$
>
> Thus, 27.75% of the power consumption can be reduced.

Chapter 2

Instruction Set Architecture (ISA)

2.1 Organization

Computer components include the processor, input, output, memory, and network. The primary focus of this course is on the processor and its interaction with the memory system. However, it is impossible to understand their operation by examining each transistor individually due to their enormous quantity. Therefore, abstraction is necessary.

Both the control unit and datapath need circuitry to manipulate instructions — for example, deciding the next instruction, decoding, and executing instructions.

There is also system software, such as the operating system and compiler, which translate programs written in high-level languages into machine instructions.

For example, after a program is written in a high-level language (like C), the compiler translates it into assembly language. Then, the assembler converts the assembly code into machine code (object code). The machine code is stored in memory, and the processor's control unit fetches an instruction from memory, decodes it to determine the operation, and signals the datapath to execute the instruction. The processor then fetches the next instruction from memory, and this cycle repeats.

2.2 Instruction Set Architecture

The instruction set architecture (ISA) is the bridge between hardware and software. It is the interface that separates software from hardware and includes all the information necessary to write a machine language program, such as instructions, registers, memory access, I/O, etc.

To put it simple, ISA is a formal specification of the instruction set that is implemented in the machine hardware. It defines how software can control the hardware by specifying the instructions, registers, memory addressing modes, and I/O operations that the processor can execute.

Assembly language instructions are the language of the machine. We aim to design an ISA that makes it easy to build hardware and compilers while maximizing performance and minimizing cost. Therefore, in this course, we focus on the RISC-V ISA.

In a Reduced Instruction Set Computer (RISC), we have fixed instruction lengths, a load-store instruction set, and a limited number of addressing modes and operations. Thus, it is optimized for speed.

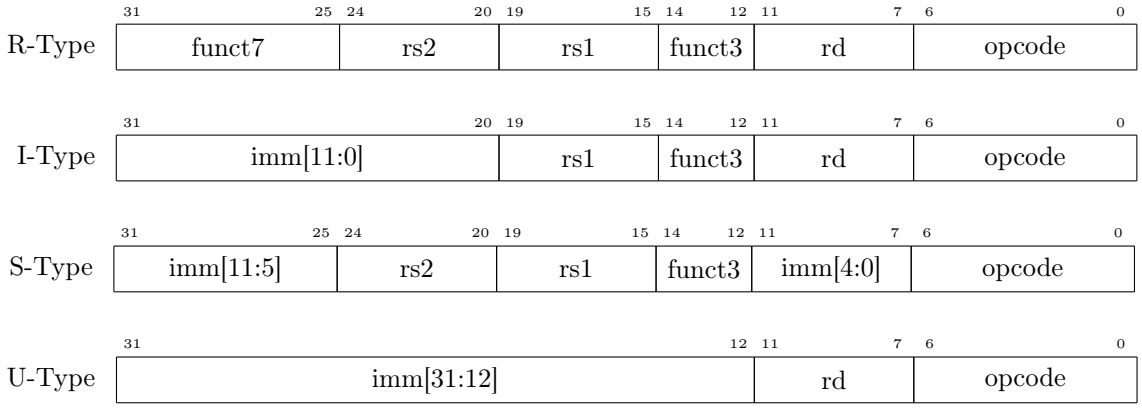
There are four design principles in RISC-V:

1. Simplicity favours regularity.
2. Smaller is faster.
3. Make the common case fast.
4. Good design demands good compromises.

2.3 RISC-V

There are five Instruction Categories:

1. Load and Store instruction
2. Bitwise instructions
3. Arithmetic instructions
4. Control transfer instructions
5. Pseudo instructions



Register names	ABI Names	Description
x0	zero	Hard-wired zero
x1	ra	Return address
x2	sp	Stack pointer
x3	gp	Global pointer
x4	tp	Thread pointer
x5	t0	Temporary / Alternate link register
x6-7	t1 - t2	Temporary register
x8	s0 / fp	Saved register / Frame pointer
x9	s1	Saved register
x10-11	a0 - a1	Function argument / Return value registers
x12-17	a2 - a7	Function argument registers
x18-27	s2 - s11	Saved register
x28-31	t3 - t6	Temporary register

Chapter 3

Arithmetic

Chapter 4

Control

Chapter 5

Logic basis

Chapter 6

Arithmetic Logic Unit

Chapter 7

Datapath

Chapter 8

Floating Number

Chapter 9

Pipeline

Chapter 10

More on Pipeline

Chapter 11

Performance

Chapter 12

Memory

Chapter 13

Cache

Chapter 14

Cache Disc

Chapter 15

Virtual Machine

Chapter 16

Instruction-Level Parallelism