Contents

Chapter 1

Micro-Architecture

Micro-Architecture is the way a given ISA is implemented on a processor. Basic blocks of a Micro-Architecture:

Cache A high-speed unit to keep code and data.

- **ICache** holds instructions.
- **DCache** holde data.

IFU A unit to fetch instruction from cache.

IDU A unit to decode instruction after fetch.

EU A unit to execute instruction:

- **ALU** for arithmetic and logic.
- **Branch Unit** for branching instruction.
- **L/S Unit** for loading and saving instruction

Register File A unit to save temporary results.

Program Counter A unit to locate next instruction.

Control Unit A unit to schedule all data movement.

1.1 Standard Benchmarks

1.1.1 Performance Metrics

Latency is the time between start and finish of a single task. The number of taks finished in a give unit of time **Throughput**. **Response time** is the total time to complete a task, also called. Response time consistes of

1. CPU time
   (a) User CPU time
   (b) System CPU time: time spent in OS doing tasks on behalf of a program.
2. I/O time

Then **System Performance** is the inverse time elapsed and **CPU performance** is the inverse user CPU time. From user perespective response time is more important and from system admin perespective throughput is more important.

For CPU time we have:

$$\begin{aligned} \text{CPU time} &= \text{CPU clock cycle} \times \text{Clock cycle time} && (1.1) \\ &= \text{CPU clock cycle/Clock rate} && (1.2) \end{aligned}$$

To have an estimation for CPU clock cycle we define **CPI**, *cycle per instruction*, to be the average clock cycles per instruction. Thus the Equations (1.1) and (1.2) become

$$\begin{aligned} \text{CPU time} &= \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time} \\ &= (\text{Instruction count} \times \text{CPI})/\text{Clock rate} \end{aligned}$$

Therefore, CPU time is dependent on:

1. Instruction count
   - Program (e.g. Algorithm)
   - ISA
   - Compiler
2. CPI
   - μArch
   - Code CPI also depends on program
3. Clock cycle time
   - μArch and technology

TO evaluate the performance of a computer we use **benchmarks**.

- SPEC Benchmarks
  - CPU/Memory intensive benchmarks
  - some stress CPU
  - some stress memory subsystem
- SPEC Web
  - I/O intensive benchmarks
  - mostly stress I/O subsystem

1.1.2 Speedup

$$\text{Speedup} = \text{Time}_{\text{original}}/\text{Time}_{\text{improved}}$$

Amdahl's law states that speedup is limited to the improved fraction. That is

$$\begin{aligned}\text{Time}_{\text{improved}} &= \text{Time}_{\text{original}} \times \left[(1 - \text{Fraction}_{\text{improved}}) + \text{Fraction}_{\text{improved}}/\text{Speedup}_{\text{improved}}\right] \\ \text{Speedup}_{\text{overall}} &= \text{Time}_{\text{original}}/\text{Time}_{\text{improved}} \\ &= \frac{1}{(1 - \text{Fraction}_{\text{improved}}) + \text{Fraction}_{\text{improved}}/\text{Speedup}_{\text{improved}}}\end{aligned}$$

MIPS, *million instruction per second*, is a performance metric. The drawbacks of this metric are

1. not taking into account capabilites of instructions
2. not realistic metric even on the same CPU

FLOPS, *floating point operations per second*, is another performance metric that measures floating point operations per unit time.

$$\text{FLOPS} = (\text{FP ops/program}) \times (\text{program/time})$$

The drawbacks of this metric are

1. ignores other instruction (e.g. load/store)
2. not all floating point operations have common format
3. depends on how FP-instensive program is

1.2 RTL

Data movement is characterized in terms of

- Registers
- Operations done on registers

RTL, *Register Transfer Language*, describes data movement at register level.

Micro-operations is the functions performed on registers. For example, shift, clear, load, increment, etc. One can describe a μArch in terms of RTL and μOps and control signals that initiate sequence of μOps to perfrom functions.

1.2.1 Register Transfer

Point-to-Point each register has dedicated wires and MUX.

Common input from MUX each register has a load enable and result of the MUX goes to each registe.

Multiple busses something between Common input and point-to-point.

Common bus with output enable similar to common input.

1.2.2 Memomry Transfer

We have **MER**, *Memory Address Register* which enables us to access memory.