UNIVERSITI MALAYSIA PERLIS
EKT303-Principle Computer Architecture
Assignment 1: Sem 2 2019-2020

NAME : Mohd Khairul Asyraff Bin Abdul Latiff
NO MATRIC : 181021080
RK : Rk 20- Computer Engineering

Question 1

(C2, C02, PO2)

a. In determining the performance of a computer system, two basic measurement metrics are used. Define the latency and throughput in term computer performance.

(3 Marks)

- **Latency** is the number of processor clocks it takes for an instruction to have its data available for use by another instruction.
- **Latency can** be said time to finish a fixed task.
- **Throughput** is the number of processor clocks it takes for an instruction to execute or perform its calculations.
- **Throughput** also the number of tasks in fixed time.
- Different: exploit parallelism for **throughput**, not **latency**.

(C6, C02, PO2)

b. As Computer Engineering, you are asked to choose a multicore system for your company. Currently, you are using a single-core machine which costs RM 56. You realize that the main application you run on the system is 75% parallelizable so the company decides to upgrade to multicore. You are given the following three options:
   - Purchase a 4-core system which costs RM 450
   - Purchase a 16-core system which costs RM 1,200
   - Purchase a 32-core system which costs RM 3,500

(10 Marks/)

It can be solved by Amdahl's law for finding Speedup due to parallelization.

By Amdahl's law, speed up of any System is,

Amdahl's law:

$$S(n) = \frac{1}{(1-P) + \frac{P}{n}}$$

S(n) = Theoretical speed-up

P = Fraction of the program that can be be made parallel

n = Number of CPU cores

Here, P = 0.75 because 75% program is parallelizable

For Single core Processor,

S(1) = 1 for RM 56

Now, Putting N = 4, 16 and 32 subsequently,

for 4-core system

P = 75% = 0.75;
n = 4 cores;

From amdahl's formula,

S(4) = 1/((1-0.75) + (0.75 / 4))

S(4) = 1/(0.25 + 0.188)

S(4) = 1/0.438 = 2.283

S(4) = 2.28*100 = 228% for RM 450

Speedup achieved = 2.28 times more of a Single-core Processor for 450-56 = RM 394 extra,

Therefore, Per Unit RM Speed achieved = 228/394 = 0.57%

for 16-core system

P = 75% = 0.75;
n = 16 cores;

From amdahl's formula,

S (16) = 1/ ((1-0.75) + (0.75 / 16))

S (16) = 1/ (0.25 + 0.047)

S (16) = 1/0.297 = 3.367

S(16) = 3.36*100 = 336% for RM 1200

Speedup achieved = 3.36 times more than a Single-core Processor for 1200-56 = RM 1144 extra,

Therefore, Per Unit RM Speed achieved = 336/1144 = 0.29%

for 32-core system

P = 75% = 0.75;
n = 32 cores;

From amdahl's formula,

S (32) = 1/ ((1-0.75) + (0.75 / 32))

S (32) = 1/ (0.25 + 0.023)

S (32) = 1/0.273 = 3.663

S(32) = 3.65*100 = 365% for RM 3500

Speedup achieved = 3.65 times more than a Single-core Processor for 3500-56 = RM 3444 extra, Therefore, Per Unit RM Speed achieved = 365/3444 = 0.10%

Therefore, We get most value for money offer by buying 4-core system that costs 0.57% speedup per RM spent.

c. Consider two different implementations, M1 and M2, of the same instruction set. There are three classes of instructions (A, B, and C) in the instruction set. M1 has a clock rate of 400 MHz, and M2 has a clock rate of 200 MHz. The average number of cycles for each instruction, or CPI, for each class of instruction on M1 and M2, is given in the following table:

Class	CPI on M1	CPI on M2	C1 usage	C2 usage
A	4	2	30%	30%
B	6	4	50%	20%
C	8	3	20%	50%

The table also contains a summary of how two different compilers use the instruction set. C1 is a compiler produced by the makers of M1 and C2 is a compiler produced by the makers of M2. Assume that each compiler uses the same number of instructions for a given program but that the instruction mix is as described in the table.

i. Using C1 on both M1 and M2, how much faster can the makers of M1 claim that M1 is compared with M2?

(6 Marks)

For C1 in M1: We have

class	CPI on M1	C1 usage
A	4	30%
B	6	50%
C	8	20%

So average CPI on M1 using compiler C1 will be:
avg CPI = 30% of 4 + 50% of 6 + 20% of 8
= 1.2 + 3 + 1.6
avg CPI on M1 using compiler C1 = 5.8
provided M1 has a clock rate of 400 MHz
So, clock cycle time of M1 is = 1/400 MHz

*CPU execution time = number of instructions executed * avg CPI * clock cycle time*

next, putting values in formula (I am taking number of instructions = 1)

execution time = 1 * 5.8 * 1/400 * 10 ^ (-6) seconds
= 5.8 * 0.0025 * 10 ^ (-6) seconds
= 0.0145 * 10 ^ (-6) seconds
= 1.45 * 10 ^ (-8) seconds

For C1 in M2:

Class	CPI on M2	C1
A	2	30%
B	4	50%
C	3	20%

So average CPI on M2 using compiler C1 will be:
avg CPI = 30% of 2 + 50% of 4 + 20% of 3
= 0.6 + 2 + 0.6
avg CPI on M2 using compiler C1 = 3.2
provided M2 has a clock rate of 200 MHz
So clock cycle time of M2 is = 1/200 MHz

*CPU execution time = number of instructions executed * avg CPI * clock cycle time*

next, putting values in formula (I am taking number of instructions = 1)

execution time = 1 * 3.2 * 1/200 * 10 ^ (-6) seconds
= 3.2 * 0.005 * 10 ^ (-6) seconds
= 0.016 * 10 ^ (-6) seconds
= 1.6 * 10 ^ (-8) seconds

From above calculated CPU execution time we have:
execution time of M1 with C1 = **1.45 * 10^ (-8) sec**
execution time of M2 with C1 = **1.6 * 10^ (-8) sec**

Speed up of Machine architecture M1 compared with M2
= (1.6 - 1.45) * 10^ (-8) sec
= 0.15 * 10^ (-8) seconds
So, M1 using C1 takes 0.15 * 10^ (-8) less seconds as compared to M2 using C1

ii. Using C2 on both M1 and M2, how much faster can the makers of M2 claim that M2 is compared with M1?

(6 Marks)

For C2 in M1:

class	CPI on M1	C2 usage
A	4	30%
B	6	20%
C	8	50%

So average CPI on M1 using compiler C2 will be:
avg CPI = 30% of 4 + 20% of 6 + 50% of 8
= 1.2 + 1.2 + 4
avg CPI on M1 using compiler C2 = 6.4
provided M1 has a clock rate of 400 MHz
So, clock cycle time of M1 is = 1/400 MHz

*CPU execution time = number of instructions executed * avg CPI * clock cycle time*

next, putting values in formula (I am taking number of instructions = 1)

Execution time = 1 * 6.4 * 1/400 MHz
= 6.4 * 0.0025 * 10 ^ (-6) seconds
= 0.016 * 10 ^ (-6) seconds
= 1.6 * 10 ^ (-8) seconds

For C2 in M2: We have

Class	CPI on M2	C2
A	2	30%
B	4	20%
C	3	50%

So average CPI on M2 using compiler C2 will be:
avg CPI = 30% of 2 + 20% of 4 + 50% of 3
= 0.6 + 0.8 + 1.5
avg CPI on M2 using compiler C2 = 2.9
provided M2 has a clock rate of 200 MHz
So, clock cycle time of M2 is = 1/200 MHz

*CPU execution time = number of instructions executed * avg CPI * clock cycle time*

next, putting values in formula (I am taking number of instructions = 1)

Execution time = 1 * 2.9 * 1/200 MHz
= 2.9 * 0.005 * 10 ^ (-6) seconds
= 0.0145 * 10 ^ (-6) seconds
= 1.45 * 10 ^ (-8) seconds

From above calculated CPU execution time we have:
execution time of M1 with C2 = **1.6 * 10 ^ (-8) sec**
execution time of M2 with C2 = **1.45 * 10 ^ (-8) sec**

Speed up of Machine architecture M1 compared with M2
= (1.6 - 1.45) * 10 ^ (-8) sec
= 0.15 * 10 ^ (-8) seconds
So, M2 using C2 takes 0.15 * 10 ^ (-8) less seconds as compared to M1 using C2

M2 is faster

Question 2

(C5, C03, PO3)

a. A computer system needs 256 bytes of RAM, 1024 x 16 of ROM and two I/O interface modules with 256 registers each with each register size is 8-bits. Build the memory system of the new computer using a 128 x 8 RAM and 512 x 8 ROM.

i. Compute the total number of bits required to address the memory system.

S.NO	Memory	N x W	N^1 x W^1	P	q	p * q	x	y	z	Total
1	RAM	128 x 8	256 x 8	2	1	2	7	1	2	10
2	ROM	512 x 8	1024 x 16	2	2	4	9	1	2	12
3	Interface	256		2	1	2	8	1	2	11
4										

Requirements:

- q is 1 always for interfaces.
- Number of registers = 2x
- P = number of interfaces
- Number of data lines = size of registers

ii. Design the memory-address map for the above system.

(10 Marks/Markah)

Memory Address Map

Component	Hexadecimal Address		Address Bus													
	From	To	15 - 12	11	10	9	8	7	6	5	4	3	2	1	0
RAM1	0000	007F		0	0	0			x	x	x	x	x	x	x
RAM2	0200	027F		0	0	1			x	x	x	x	x	x	x
ROM1.1	0400	05FF		0	1	0	x	x	x	x	x	x	x	x	x
ROM1.2	0400	05FF		0	1	0	x	x	x	x	x	x	x	x	x
ROM2.1	0600	07FF		0	1	1	x	x	x	x	x	x	x	x	x
ROM2.2	0600	07FF		0	1	1	x	x	x	x	x	x	x	x	x
Interface1	0800	08FF		1	0	0		x	x	x	x	x	x	x	x
Interface2	0A00	0AFF		1	0	1		x	x	x	x	x	x	x	x