Homework Assignment 1

CS/ECE 6810: Computer Architecture

Jake Pitkin u0891770

Performance Metrics, ISA, Basic Pipelining

September 12, 2018

1. **Performance Optimization.** The table below shows the frequencies and cycle counts for all types of instructions used by program A on a computer system P. We observed that 50% of the executed MULT instructions are followed by an ADD. Therefore, we propose a new processor with a fused-MULT-ADD (FMAD) instruction that executes a merged MULT and ADD in 5 cycles.

	LOAD	STORE	BRANCH	ADD	MULT
Frequency	15%	15%	10%	20%	40%
Cycles	1	1	2	4	5

   i Compute the instructions per cycle (IPC) for the new and old processors. **(10 points)**

   Old processor: First we will calculate the cycles per instruction (CPI) and take the multiplicative inverse to find ICP.

$$\begin{aligned}\text{CPI} &= \sum_{i=1}^{n} I_{freq} * I_{cycles} \\ &= 0.15 * 1 + 0.15 * 1 + 0.1 * 2 + 0.2 * 4 + 0.4 * 5 \\ &= 3.3\end{aligned}$$

$$\text{Old ICP} = \frac{1}{\text{Average CPI}} = \frac{1}{3.3} = \boxed{0.30303}$$

   New processor: Next we will add the FMAD instruction to the instruction set. We will replace 50% of the executed MULT instructions, and the ADD that follows them, with FMAD operations.

	LOAD	STORE	BRANCH	ADD	MULT	FMAD
Frequency	15%	15%	10%	0%	20%	20%
Cycles	1	1	2	4	5	5

   We need to renormalize the table by dividing each frequency by the sum of the frequencies.

	LOAD	STORE	BRANCH	ADD	MULT	FMAD
Frequency	18.75%	18.75%	12.5%	0%	25%	25%
Cycles	1	1	2	4	5	5

   Using this table we can calculate the ICP as we did above:

$$\begin{aligned}\text{CPI} &= \sum_{i=1}^{n} I_{freq} * I_{cycles} \\ &= 0.1875 * 1 + 0.1875 * 1 + 0.125 * 2 + 0.25 * 5 + 0.25 * 5 \\ &= 3.125\end{aligned}$$

$$\text{New ICP} = \frac{1}{\text{Average CPI}} = \frac{1}{3.125} = \boxed{0.32}$$

ii What is the speedup/slowdown gained through the proposed optimization? **(5 points)**

First we must calculate the CPU time for both the old and new processor. The CPU time for the old processor is given by :

$$\text{CPU Time} = IC * CPI * CT$$

We aren't given the clock cycle time (CT) but we know it is the same for both processors. We calculated the IC (the new processor reduced it's instruction count by 20% with FMAD) and CPI for both processors in part i which we will use here.

$$\begin{aligned}\text{Old CPU Time} &= IC_o * CPI_o * CT_o \\ &= 1 * 3.3 * CT\end{aligned}$$

$$\begin{aligned}\text{New CPU Time} &= IC_n * CPI_n * CT_n \\ &= 0.8 * 3.125 * CT \\ &= 2.5\end{aligned}$$

$$\begin{aligned}\text{speedup} &= \text{old CPU Time/new CPU Time} \\ &= (3.3 * CT)/(2.5 * CT) \\ &= \boxed{1.32}\end{aligned}$$

The new processor provides a speedup of 132% compared to the old processor.

2. **Execution Time.** A computer system has three instruction classes as shown in the following table.

Class	CPI
A	1
B	2
C	3

We execute two programs (P1 and P2) on this computer system. The following table shows the number of executed instructions for different classes.

Program	A	B	C
P1	200	100	200
P2	400	100	100

i What is the CPI and IPC of P1 and P2? **(10 points)**

We can calculate CPI and IPC using the following formulas:

$$\text{CPI} = \frac{\text{CPU clock cycles for a program}}{\text{Instruction Count}}$$

$$\text{IPC} = \frac{1}{\text{CPI}}$$

Program P1:

$$\begin{aligned}\text{P1 CPI} &= \frac{(1*200)+(2*100)+(3*200)}{200+100+200} \\ &= \frac{1000}{500} \\ &= \boxed{2}\end{aligned}$$

$$\begin{aligned}\text{P1 IPC} &= \frac{1}{\text{P1 CPI}} \\ &= \frac{1}{2} \\ &= \boxed{0.5}\end{aligned}$$

Program P2:

$$\begin{aligned}\text{P2 CPI} &= \frac{(1*400)+(2*100)+(3*100)}{400+100+100} \\ &= \frac{900}{600} \\ &= \boxed{1.5}\end{aligned}$$

$$\begin{aligned}\text{P2 IPC} &= \frac{1}{\text{P1 CPI}} \\ &= \frac{1}{1.5} \\ &= \boxed{0.667}\end{aligned}$$

ii Suppose that the computer operates at 1GHz clock frequency; compute the execution times of P1 and P2? **(10 points)**

We can calculate CPU time using the following formulas:

$$\text{CPU Time} = \text{IC} * \text{CPI} * \text{CT}$$

Program P1: We know all the values needed to calculate P1's CPU time:

$$\begin{aligned}\text{P1 IC} &= 200+100+200 = 500 \\ \text{P1 CPI} &= 2 \text{ (from part i)} \\ \text{P1 CT} &= \frac{1}{1GHz} = \frac{1}{1,000,000,000}\end{aligned}$$

$$\begin{aligned}\text{P1 CPU Time} &= IC*CPI*CT \\ &= \frac{500*2}{1,000,000,000} \\ &= 0.0000001 \\ &= \boxed{1{,}000 \text{ ns}}\end{aligned}$$

Program P1: Similar to P1, we can calculate the CPU time:

$$\begin{aligned}\text{P2 IC} &= 400 + 100 + 100 = 600\\ \text{P2 CPI} &= 1.5 \text{ (from part i)}\\ \text{P2 CT} &= \frac{1}{1GHz} = \frac{1}{1,000,000,000}\end{aligned}$$

$$\begin{aligned}\text{P2 CPU Time} &= IC * CPI * CT\\ &= \frac{600 * 1.5}{1,000,000,000}\\ &= 0.0000009\\ &= \boxed{900 \text{ ns}}\end{aligned}$$

3. **Power and Energy.** A general purpose processor is operated at a 2GHz clock frequency with a 5V voltage supply. Suppose that the average load capacitance of all internal gates is 12pF. The static current driven from the supply is 5A and the logic gates exhibit an average activity of 0.8 switchings per cycle.

   i Compute the static and dynamic power of the processor. **(10 points)**

   We can compute static and dynamic power using the following formulas:

$$Power_{Static} = Voltage * Current_{Static}$$

$$Power_{Dynamic} \propto Capacitance * Voltage^2 * (Activity * Frequency)$$

   Static power: We are given the voltage of the processor is 5V and the static current is 5A.

$$\begin{aligned}Power_{Static} &= Voltage * Current_{Static}\\ &= 5V * 5A\\ &= \boxed{25\text{W}}\end{aligned}$$

   Dynamic Power: We are given the capacitance is 12pF, the voltage is 5V, the activity is 0.8, and the frequency 2GHz.

$$\begin{aligned}Power_{Dynamic} &\propto Capacitance * Voltage^2 * (Activity * Frequency)\\ &= 12pF * 5V^2 * (0.8 * 2GHz)\\ &= 12 * 10^{-12}\frac{A * sec}{V} * 5V * 5V * (1.6 * 10^9\frac{switch}{sec})\\ &= 300 * 10^{-12}A * V * (1.6 * 10^9)\\ &= \boxed{0.48\text{W}}\end{aligned}$$

   ii If the voltage is scaled down by 50% and the frequency is scaled up by 25%, calculate the percentage increase/decrease in the total system power. **(10 points)**

   Scaling the voltage down 50% gives a voltage of 2.5V and scaling frequency up 25% gives a frequency of 2.5GHZ.

$$
\begin{aligned}
New\ Power_{Static} &= Voltage * Current_{Static} \\
&= 2.5V * 5A \\
&= 12.5W \\
New\ Power_{Dynamic} &\propto Capacitance * Voltage^2 * (Activity * Frequency) \\
&= 12pF * 2.5V^2 * (0.8 * 2.5GHz) \\
&= 12 * 10^{-12} \frac{A * sec}{V} * 2.5V * 2.5V * (2 * 10^9 \frac{switch}{sec}) \\
&= 75 * 10^{-12} A * V * (2 * 10^9) \\
&= 0.15W \\
New\ Power &= Power_{Static} + Power_{Dynamic} \\
&= 12.5W + 0.15W \\
&= 12.65W
\end{aligned}
$$

$$
\begin{aligned}
Old\ Power &= 25W + 0.48W \\
&= 25.48W
\end{aligned}
$$

$$
\begin{aligned}
percentage\ decrease &= 1 - \frac{12.65W}{25.48W} \\
&= 1 - 0.4964 \\
&= \boxed{50.36\% \text{ decrease}}
\end{aligned}
$$

iii If the frequency is scaled down by 50% of its initial value and the dynamic power along with the output current remains the same, calculate the percentage increase/decrease in the total system power. **(10 points)**

If the frequency is scaled down by 50% we have a new frequency of 1GHz and the dynamic power, capacitance (from Canvas discussion), activity (from Canvas discussion), and output current remains the same. To calculate the new power, we need to solve for the unknown which is the voltage.

$$
\begin{aligned}
Power_{Dynamic} &= C * V^2 * (\alpha * f) \\
0.48W &= 12pF * V^2 * (0.8 * 1GHz) \\
V^2 &= \frac{0.48W}{12pF * (0.8 * 1GHz)} \\
V^2 &= \frac{0.48}{0.0096} \\
V^2 &= 50 \\
V &= 7.07
\end{aligned}
$$

Given voltage is 7.07, we can solve for the static power of the new processor as well as the total power. With this we can calculate the increase or decrease in power.

$$
\begin{aligned}
New\ Power_{Static} &= Voltage * Current_{Static} \\
&= 7.07V * 5A \\
&= 35.35W
\end{aligned}
$$

$$
\begin{aligned}
New\ Power &= 35.35W + 0.48W \\
&= 35.83W
\end{aligned}
$$

$$
\begin{aligned}
percentage\ increase &= 1 - \frac{35.83}{25.48W} \\
&= 1 - 1.406 \\
&= \boxed{40.6\%\ \text{increase}}
\end{aligned}
$$

4. **Instruction Set Architecture.** Initial values of the architectural registers and memory are given below in the following tables. Compute the effective address and final the result for each of the following instructions. *All instructions are executed serially.* Register value changes are considered when moving from one instruction to another. List the final values of all the registers. **(20 points)**

R0	R1	R2	R3	R4	R5	R6
1000	50	2000	5000	0	0	0

Address	1000	2000	3000	4000	5000
Value	1000	4000	3000	5000	2000

`LOAD R5, 3000(R0)`

Effective Address of 3000(R0): Mem[3000 + R0] = Mem[3000 + 1000] = Mem[4000] = 5000
Operation: R5 = Mem[3000 + R0]

R0	R1	R2	R3	R4	R5	R6
1000	50	2000	5000	0	**5000**	0

`ADD R4, @(R5)`

Effective Address of @(R5): Mem[R5] = Mem[5000] = 2000
Operation: R4 = R4 + Mem[R5]

R0	R1	R2	R3	R4	R5	R6
1000	50	2000	5000	**5000**	5000	0

`SUB R2, R4`
Operation: R2 = R2 - R4 (Assuming this is the def of SUB with two args)

R0	R1	R2	R3	R4	R5	R6
1000	50	**-3000**	5000	5000	5000	0

`LOAD R6, 1000(R0)`
Effective Address of 1000(R0): Mem[1000 + R0] = Mem[1000 + 1000] = Mem[2000] = 4000
Operation: R6 = Mem[1000 + R0]

R0	R1	R2	R3	R4	R5	R6
1000	50	-3000	5000	5000	5000	**4000**

`ADD R6, R2`
Operation: R6 = R6 + R2

R0	R1	R2	R3	R4	R5	R6
1000	50	-3000	5000	5000	5000	**1000**

`SUB R5, R6`
Operation: R5 = R5 - R6 (Assuming this is the def of SUB with two args)

R0	R1	R2	R3	R4	R5	R6
1000	50	-3000	5000	5000	**4000**	1000

`ADD R2, R5`
Operation: R2 = R2 + R5

R0	R1	R2	R3	R4	R5	R6
1000	50	**1000**	5000	5000	4000	1000

`ADD R2, R3`
Operation: R2 = R2 + R3

Final Values

R0	R1	R2	R3	R4	R5	R6
1000	50	**6000**	5000	5000	4000	1000

5. **Pipelining.** We design a simple microprocessor core that implements five operations similar to those of the basic MIPS architecture. The delays of these operations are 36ns, 39ns, 23ns, 28ns and 64ns. Consider designing both non-pipelined and pipelined versions of the microprocessor. (Inserting a pipeline register between operations adds 1ns to the critical path delay.) What is the maximum achievable speedup of the pipelined version over non-pipelined architecture when executing a large number of instructions (n>>5)? **(15 points)**

   Non-pipelined version: The non-pipelined version will have a cycle time of 190ns, require 1 cycle per instruction and and ran on n instructions:

$$\begin{aligned} CPI_{NP} &= 1 \\ CT_{NP} &= 36ns + 39ns + 23ns + 28ns + 64ns = 190ns \\ IC &= n \end{aligned}$$

$$\begin{aligned} CPU\ time_{NP} &= IC * CT_{NP} * CPI_{NP} \\ &= n * 190ns \end{aligned}$$

   Pipelined version: We can calculate the new CPU time using the formula from lecture 4 slide page 16. *Note this is the cycle time of an IDEAL pipeline and not by taking the cycle time to be the longest running operation, 64ns in this case, and adding the pipeline register time. I originally did this but then saw the Professors comments on the Canvas discussion board:*

$$\begin{aligned} CPU\ Time_P &= IC * CPI_P * (t + CT_{NP}/P) \\ &= n * 5 * (4 + 190ns/5) \\ &= n * 210ns \end{aligned}$$

   But this assuming we aren't executing instructions simultaneously which is where we will see gains with pipelining. In an ideal situation, when n is large we will be completing an instruction every cycle. So we can use the formula from lecture slide page 16 to calculate the speedup when $n >> 5$:

$$\begin{aligned} Speedup &= CT_{NP}/(t + CT_{NP}/P) \\ &= 190/(4 + 190/5)) \\ &= \boxed{4.523} \end{aligned}$$

With this ideal theoretical pipeline we can expect about 450.3% speedup as n grows large.