

HOMEWORK V

DUE : November 17, 2009

READ :

- i) Related portions of Chapter 4 (except Sections 4.4)
- ii) Related portions of Appendix B
- iii) Related portions of Appendix C
- iv) Related portions of Appendix D
- v) Related portions of Chapter 1

ASSIGNMENT : There are **eight** questions

Solve all homework and exam problems as shown in class and past exam solutions.

In calculating figures, you may have to convert **percentages** to **fractions**. Or, you may have to convert **fractions** to **percentages**.

1) Consider the EMY CPU design with nine (9) integer instructions. We decide to expand the design by adding a MIPS instruction that is already in the MIPS architecture : **ADDI** (Add Immediate). Modify the EMY CPU completely to run the ADDI instruction.

Assume that the EMY CPU is a multicycle CPU which is **microprogrammed** and executes those nine instructions in the EMY CPU Handout. You will completely modify the CPU of the handout for the **new** instruction : ADDI ! That is, you will modify the **high-level state diagram**, the **datapath**, the **low-level state diagram** and the **control unit** so that it can execute ADDI.

In order to do that you will follow steps III(b) and IV in the Digital System Design Basics handout. That is, you need to modify the **high-level state diagram** (not in terms of buses) in parallel with the modification of the **datapath**. Then, you will modify the **low-level state diagram**. If you decide to add new states, start at state 16. Finally, you will modify the **microprogrammed control unit**. You will state, if you need to change the microinstruction format and/or the hardware of the control unit. Then, you will write down the modified portion of the microcode to execute the ADDI instruction. Since, it is the microcode, you will show only 1's and 0's.

2) Consider the following CISC instruction : DECM. It decrements a memory location. The syntax, semantics, format and other information about the instruction are as follows :

- The **syntax** of the new instruction : $DECM \text{ Disp}(Rs)$

- The **semantics** of the new instruction : $M[Rs + Disp^+] \leftarrow M[Rs + Disp^+] - 1$

- The format is the **I format** :

Opcode	Rs	Rt	DOImm
6	5	5	16

Opcode = 17
Rt is **not** used

Three arguments are used by the instruction : The destination and the first source arguments are memory arguments. We use the **2-byte signed displacement** addressing mode for them. The other source argument is always $(-1)_{10}$. We use the **Implied** addressing mode for it.

Modify the EMY CPU completely ! Assume that the CPU is **microprogrammed** and executes those nine instructions in the EMY CPU Handout. It asks you to completely modify the CPU of the handout for a **new** instruction : **DECM** ! That is, you will modify the **high-level state diagram**, the **datapath**, the **low-level state diagram** and the **control unit** so that it can execute the **DECM** instruction.

You will state, if you need to change the microinstruction format and/or the hardware of the control unit. Then, you will write down the **modified** portion of the **microcode** to execute the **DECM** instruction. Since, it is the microcode, you will show only 1's and 0's. If you decide to add new states, start at state 16.

3) A program is run on machines M1 and M2 and the following execution statistics is obtained :

Time on Machine 1	Time on Machine 2	Instructions executed on M1	Instructions executed on M2	Clock frequency of M1	Clock frequency of M1
2 seconds	1.5 seconds	5×10^9	6×10^9	4 GHz	6 GHz

Calculate the following :

i) The **MIPS_{ave}** figures for the program on Machine M1 and on Machine M2.

ii) The **CPI_{ave}** figures for the two machines when they run the program.

4) Solve Problem 1.14.1 of Chapter I.

You are asked to compute **CPUtime** figures for the program on **each** processor. That is, you will obtain **two** CPUtime figures to compare the performance of the two processors.

Note that CPI_i figures are very low. This is because **pipelining** is used to speed up the execution !

5) Solve Problem 1.14.4 of Chapter I.

You are asked to compute $\text{MFLOPS}_{\text{ave}}$ figures for the program on **each** machine. That is, you will obtain **two** $\text{MFLOPS}_{\text{ave}}$ figures to compare the performance of two machines.

To calculate the MFLOPS figures, you will calculate the **FP execution time** figures for the two programs. This is possible since the question specifies the CPI_{FP} figures for the machine.

6) Solve Problem 1.14.5 of Chapter I.

You are asked to compute MIPS_{ave} figures for the program on **each** machine. That is, you will obtain **two** MIPS_{ave} figures to compare the performance of two machines.

Note that to calculate the MFLOPS figures, you will also calculate the **CPUtime** figures for the two programs. This is possible since the question specifies separate CPI figures for different types of instructions for the machine.

7) Solve Problem 1.14.6 of Chapter I.

To solve the question you will use CPUtime to calculate the Performance where $1/\text{CPUtime}$ is the Performance. The question is asking you to mention if there is any correlation between Performance and MFLOPS and any correlation between Performance and MIPS.

8) Assume that a new processor is developed. The design of the processor and compiler are completed, and a decision has to be made whether to produce the current processor design as it stands or spend additional time to improve it. There are two options :

Option a :

Leave the design as it stands. Let's call this base computer, Mbase. It has a clock rate of 500 MHz, and the following measurements have been made using a simulator:

Instruction Class	CPI_i	Frequency
A	2	40%
B	3	25%
C	3	25%
D	5	10%

Option b :

Optimize the hardware so that the clock rate is increased to 600 MHz. Let's call this computer, Mopt. The following measurements were made using a simulator for Mopt:

Instruction Class	CPI _i	Frequency
A	2	40%
B	2	25%
C	3	25%
D	4	10%

What is the CPI_{ave} for each computer?

The CPI_{ave} is calculated by using :

$$\text{CPI}_{\text{ave}} = \sum_{i=1}^n \text{CPI}_i \times \text{Fraction}_i$$

Fraction_i is the conversion of Frequency_i to a fraction. For example, if the frequency is 40%, then the fraction is 0.4.

Let's try to visualize what the designers are going through :

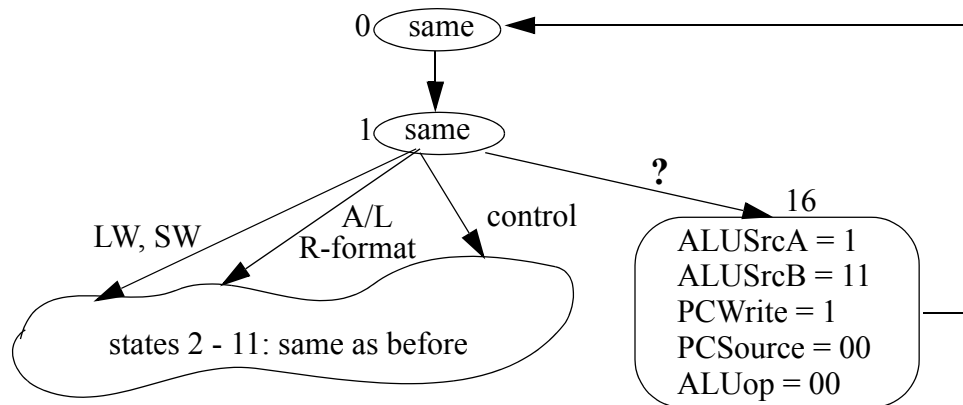
According to the tables, instructions in class **A** are the most common and must immediately be targeted for further hardware optimization. However, Class **A** instructions already take a very short time to run : Two clock periods. It would be too expensive to shorten their run time further.

The next candidates are Class **B** and **C** instructions which are more common than Class **D** instructions to improve the run time. It turns out that it is the case....

Note also that the clock frequency of machine Mopt is increased which improves the run time of **all** instruction classes : A, B, C and D.

RELEVANT QUESTIONS AND ANSWERS

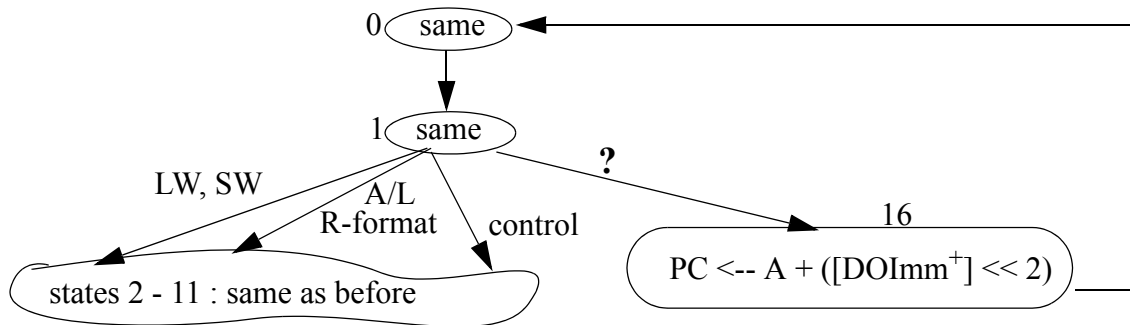
Q1) The EMY CPU is designed as shown in class. Its control unit is microprogrammed ! The CPU is modified due to an architectural change and the resulting low-level state diagram is as follows :



- > What is the corresponding high-level state diagram ?
- > Draw the modified control unit which is still microprogrammed.

Then, you will write down the modified portion of the microcode to execute the ADDI instruction. Since, it is the microcode, you will show only 1's and 0's.

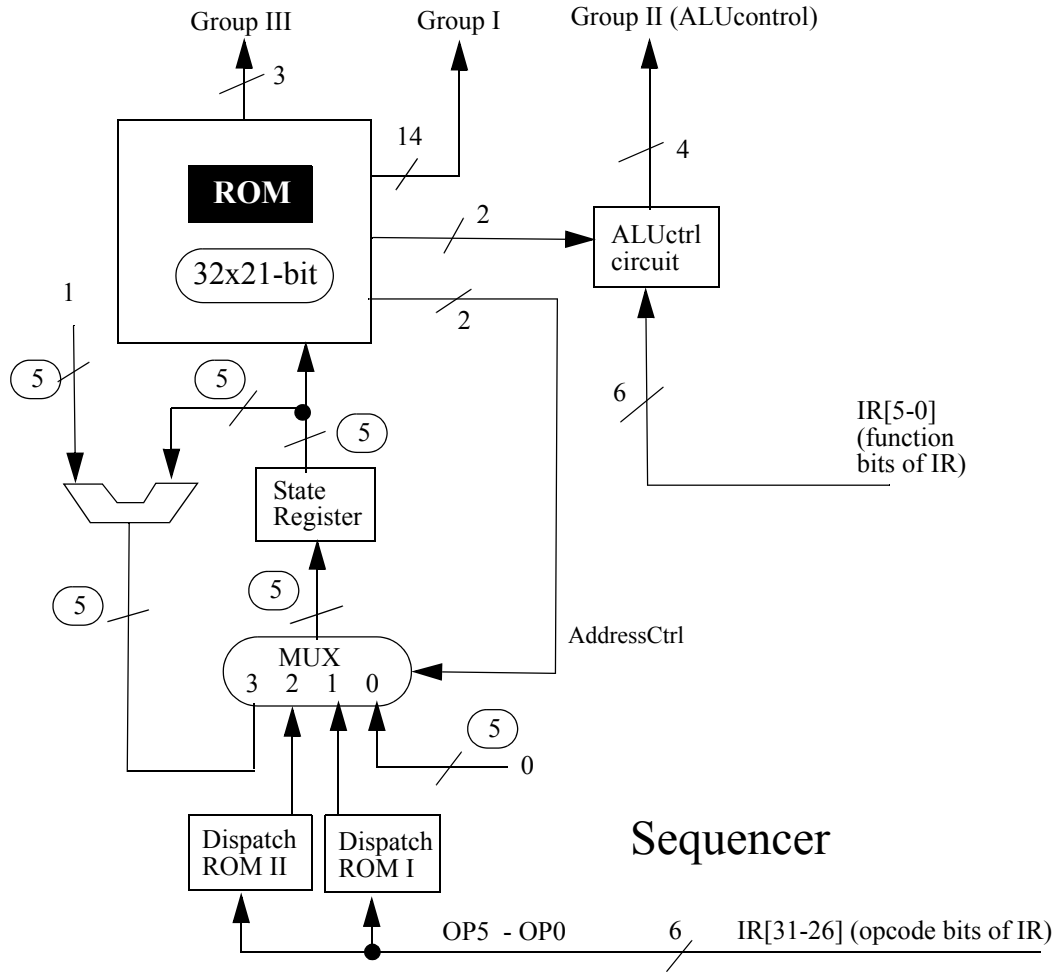
A1) The modified portion of the high-level state diagram is as follows :



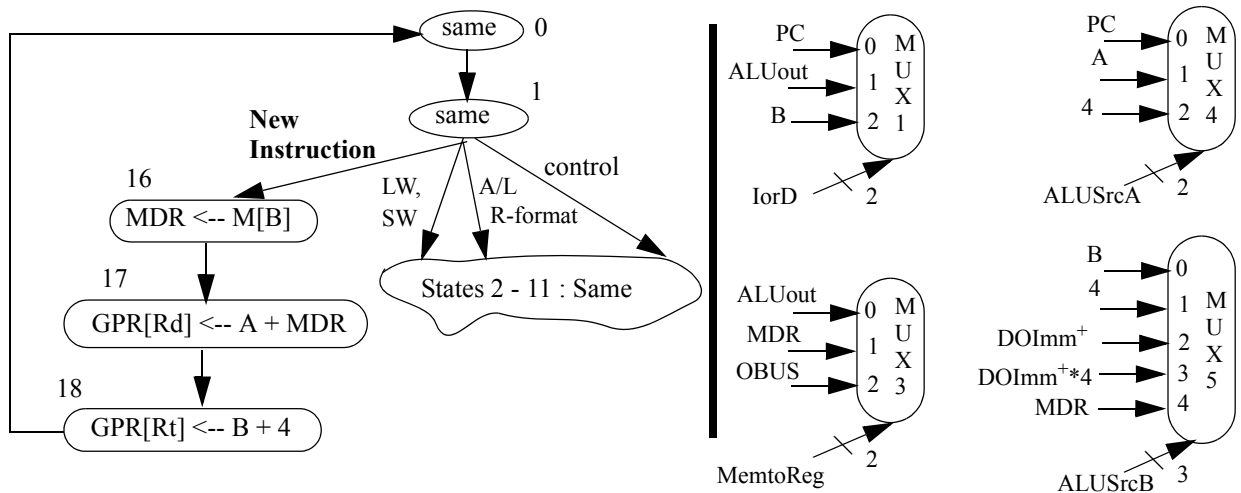
The modified sequencer is shown below. The changes on the Control Unit are due to having having a microinstruction in location $(16)_{10}$. All the changes on the sequencer are circled to quickly locate them. Dispatch ROMs are now 64x5-bit. In addition, Dispatch ROM I is programmed in a location corresponding to the opcode value with the content of $(16)_{10}$.

The ROM has in its location $(16)_{10}$ has a new content. We show the **rightmost** 18 bits of the new microinstruction below :

Loc	PCWrite	PCWCond	IorD	MemRead	MemWrite	IRWrite	MementoReg	PCSource	ALUop	ALUSrcB	ALUSrcA	RegWrite	RegDst	AddrCtrl
16	1	0	0	0	0	0	0	00	00	11	1	0	0	00



Q2) Assume that the EMY **high-level** state diagram and the **datapath** have been modified to be able to run a new instruction as shown below :



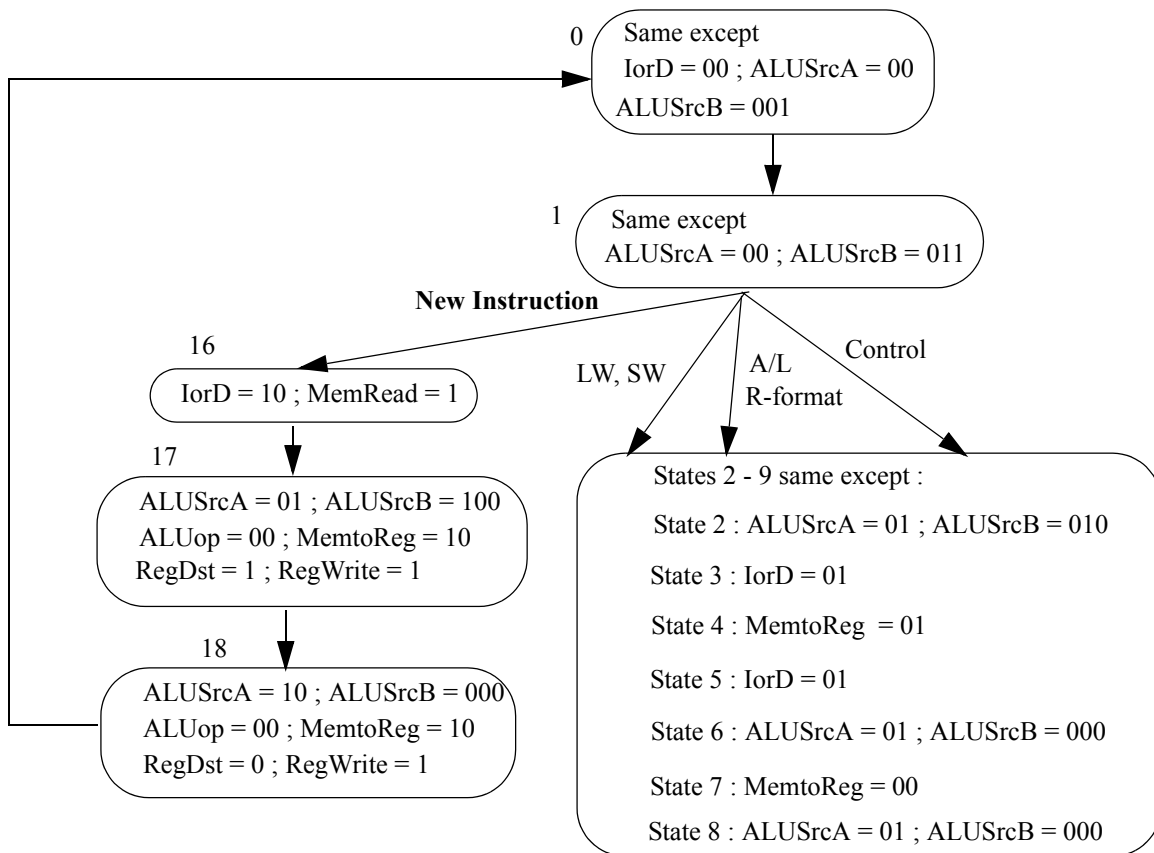
a) Modify the EMY low-level state diagram accordingly.

b) Assume that the EMY CPU is **microprogrammed**.

i) What is the **new** size of the micromemory ? Explain.

ii) Show the microcode for micromemory locations 16, 17 and 18. Are original micromemory locations different ? Why ?

A2) a) The modified low-level state diagram is as follows :



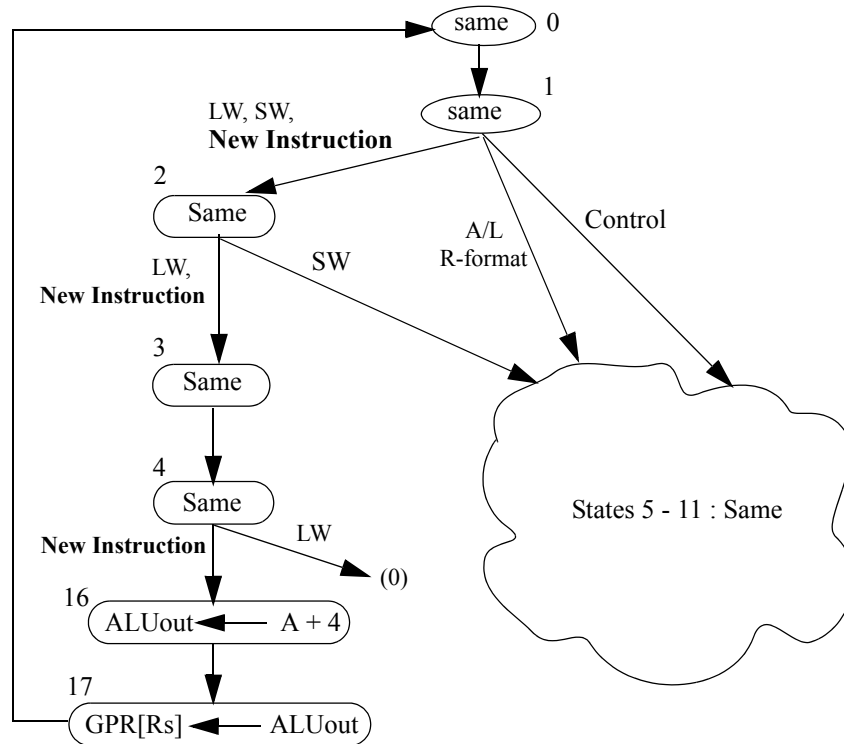
b)

i) Since we have states 16, 17 and 18, there are 32 locations in the micromemory.

We have increased the number of control signals by four : IorD, ALUSrcA, ALUSrcB and MemtoReg have one new bit each.

Then, each micromemory location has $21 + 4 = 25$ bits. Therefore, the new size is 32×25 -bit.

The new microprogrammed Control Unit is shown below. The changes on the Control Unit are due to having having new microinstructions. All the changes on the sequencer are circled to quickly locate them. Dispatch ROMs are now 64x5-bit. In addition, Dispatch ROM I is programmed in a location corresponding to the opcode value with the content of $(16)_{10}$.



a) Modify the EMY **datapath**. How long does it take to run the new instruction ?

b) Modify the EMY **low-level** state diagram accordingly.

c) Assume that the EMY CPU is **microprogrammed**.

i) What is the **new** size of the micromemory ? Explain.

ii) Show the **microcode** for micromemory locations 16 and 17.

d) Do you think the new instruction can be used for the program below ? Why ?

400000	LW	R8, 0(R9)	# R9 points at array A and initially has 10000000
400004	ADDI	R8, R8, (-1) ₁₀	
400008	OR	R8, R8, R10	# R10 is already initialized with a value
40000C	SLL	R8, R8, 2	
400010	SW	R8, 0(R9)	
400014	ADDI	R9, R9, 4	
400018	ADDI	R11, R11, (-1) ₁₀	# R11 is the loop-end counter
40001C	BNE	R11, R0, (-8) ₁₀	

A3) a) The new instruction takes **7** clock periods to run since we trace states 0, 1, 2, 3, 4, 16 and 17.

The modified datapath is as follows :

Loc	PCWrite	PCWriteCond	IorD	MemRead	MemWrite	IRWrite	MemtoReg	PCSource	ALUop	ALUSrcB	ALUSrcA	RegWrite	RegDst	AddrCtrl
16	0	0	0	0	0	0	0	00	00	01	1	0	00	011
17	0	0	0	0	0	0	0	00	00	00	0	1	10	000

d) The new instruction can be used by the program in Question 1 since it loads a memory location to GPR Rt and then adds four to register Rs so that the next data access can be performed without using an explicit ADDI instruction on Rs. Below, the new instruction is described and the program in Question 1 is rewritten ;

i) The instruction loads a memory location to register and then advances (increments) the pointer register automatically.

ii) The **syntax** of the new instruction Load Word with autoIncrement : LWI Rt, Disp(Rs)++

The **semantics** of the new instruction : Rt \leftarrow M[Rs + Disp⁺] then Rs \leftarrow Rs + 4

The format is the **I format** :

Opcode	Rs	Rt	DOImm
6	5	5	16

Five arguments are used by the instruction :

The two destination arguments are registers Rs and Rt and they are explicitly specified by the instruction and so they use the Register instruction addressing mode.

For the first architectural operation, the source is a memory argument. We use the **2-byte signed displacement** addressing mode for it.

For the second architectural operation, the source arguments are register Rs which is explicitly specified hence the Register addressing mode and a constant which is implied hence the Implied addressing mode.

The program is rewritten by using the new instructions :

```

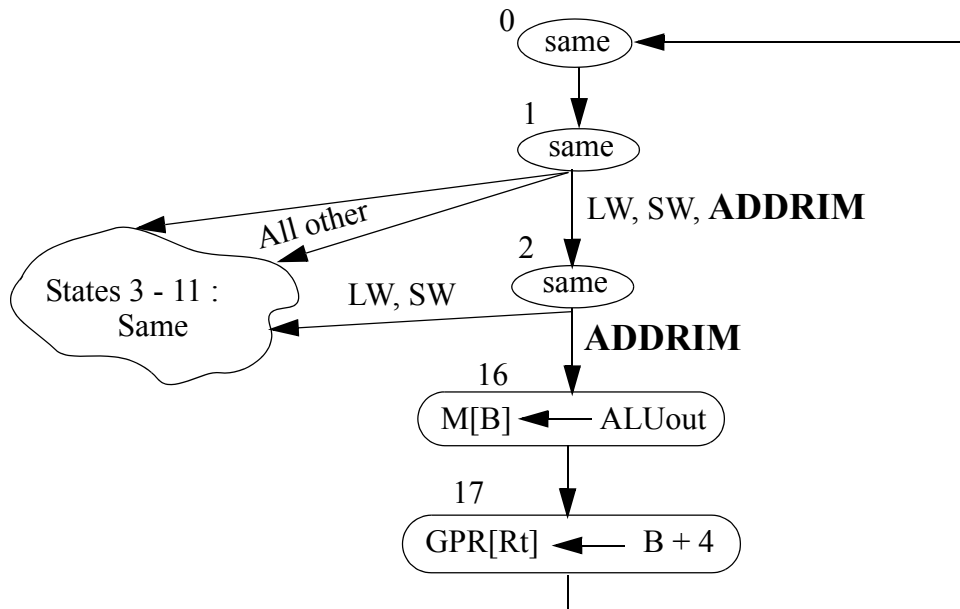
400000 LWI    R8, 0(R9)      # R9 points at array A and initially has 10000000
400004 ADDI   R8, R8, (-1)10
400008 OR     R8, R8, R10   # R10 is already initialized with a value
40000C SLL    R8, R8, 2
400010 SW     R8, 0(R9)
400014 ADDI   R11, R11, (-1)10 # R11 is the loop-end counter
400018 BNE   R11, R0, (-8)10

```

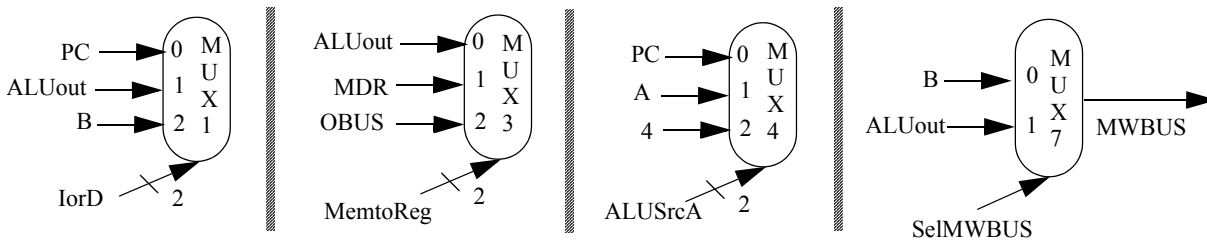
The program length is shorter. Each iteration takes 30 clock periods, not 32 as it is the case with the original program. Also, each iteration requires nine memory accesses, not 10.

However, this instruction takes 7 clock periods and pipelining it can be harder because of more stages and also a register (Rs) is read late that can cause WAR hazards.

Q4) Assume that the EMY **high-level** state diagram has been modified to be able to run a new instruction, **ADDRIM**, as shown below :



Assume that the EMY **datapath** is modified as follows :



a) Modify the EMY **low-level** state diagram accordingly.

b) Assume that the EMY CPU is **microprogrammed**.

- i) What is the **new** size of the micromemory ? **Explain**.
- ii) Show the **microcode** for micromemory locations **2, 16** and **17**.

c) This new instruction can be described as follows :

- Its syntax is : ADDRIM (Rt)++, Rs, Imm
- Its semantics is : $M[Rt] \leftarrow Rs + Imm^+$ then $Rt \leftarrow Rt + 4$

The new instruction can be used for the following program :

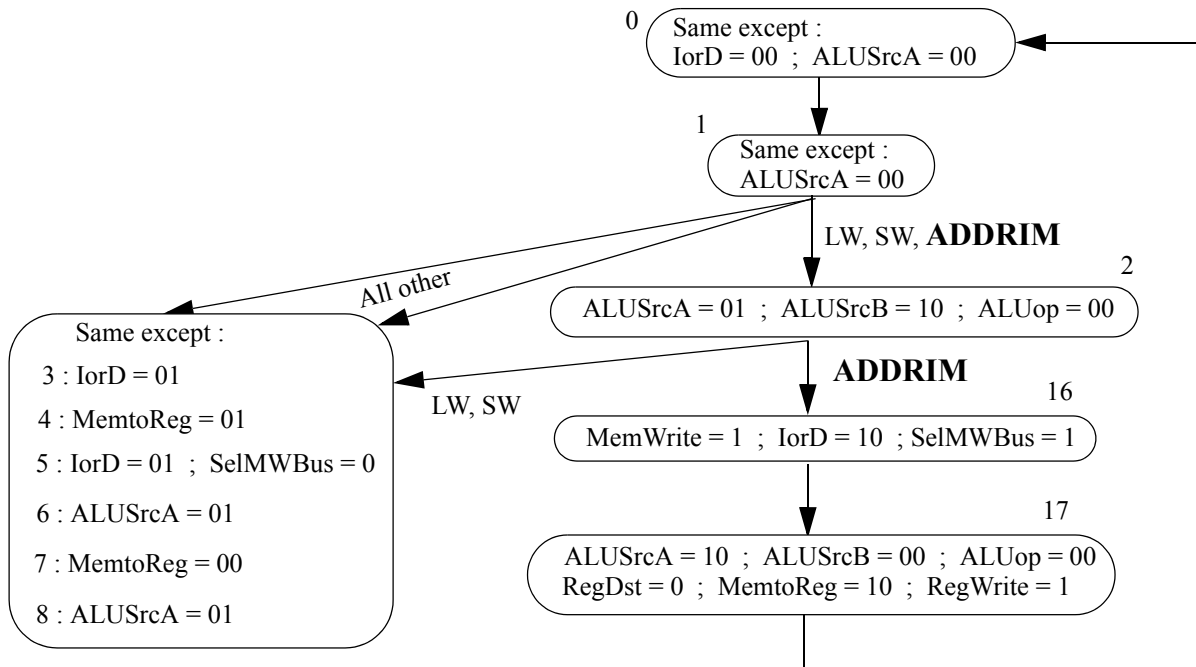
```

400C00    LW      R9, 0(R8)           # R8 points at array A and initially has 10EA0000
400C04    ADD     R9, R9, R9
400C08    ADDI   R10, R9, AC
400C0C    SW     R10, 0(R8)
400C10    ADDI   R8, R8, 4
400C14    ADDI   R11, R11, (-1)10  # R11 is the loop-end counter
400C18    BNE   R11, R0, (-7)10

```

Rewrite the program, by using the new instruction.

A4) a) The modified low-level state diagram is as follows :



b)

i) Since we have states 16, and 17, there are 32 locations in the micromemory.

We have increased the number of control signals by four. IorD, MemtoReg and ALUSrcA have one extra bit each. There is also a new control signal : SelMWBUS. Then, each micromemory location has $21 + 4 = 25$ bits. Therefore, the new size is 32×25 -bit

ii) The microcode for micromemory locations 2, 16 and 17 are as follows :

Loc	SelMWBUS	PCWrite	PCWriteCond	IorD	MemRead	MemWrite	IRWrite	MemtoReg	PCSource	ALUOp	ALUSrcB	ALUSrcA	RegWrite	RegDst	AddrCtrl
2	0	0	0	00	0	0	0	00	00	00	10	01	0	0	00
16	1	0	0	10	0	1	0	00	00	00	00	00	0	0	11
17	0	0	0	00	0	0	0	10	00	00	00	10	1	0	00

d) The program is rewritten by using the new instructions :

```

400C00  LW      R9, 0(R8)           # R8 points at array A and initially has 10EA0000
400C04  ADD     R9, R9, R9
400C08  ADDRIM (R8)++, R9, AC
400C0C  ADDI   R11, R11, (-1)10      # R11 is the loop-end counter
400C10  BNE   R11, R0, (-5)10

```

The program length is shorter. Each iteration takes 21 clock periods, not 28 as it is the case with the original program. Also, each iteration requires seven memory accesses, not nine.

However, this instruction is a CISC instruction, accessing the memory and performing an A/L instruction. Pipelining can be harder.

Q5) The following piece of mnemonic machine language program is run on the EMY computer :

```

400100 LW    R8, 0(R9)
400104 ADDI  R8, R8, 1
400108 SW    R8, 0(R9)
40010C ADDI  R9, R9, 4
400110 ???           # Decrement R10 by 1
400114 ???           # If R10 is not equal to zero, branch to location 400100

```

a) Based on the comments made, specify the last two instructions precisely.

b) The clock frequency is 200 MHz. Initially for the program, registers R9 and R10 contain 10005000 and $(100)_{10}$, respectively. The CPI_i of each instruction is as follows :

Instruction	Traced states and CPI_i
LW	0, 1, 2, 3, 4 => 5
SW	0, 1, 2, 5 => 4
ADDI	0, 1, 16, 17 => 4
M[110]	? => 4
M[114]	? => 3

Calculate how long it will take to run the above code. Also, calculate : the CPI_{ave} and the $MIPS_{ave}$

A5) a)

```

400110 ADDI  R10, R10, (-1)10      # Decrement R10 by 1
400114 BNE   R10, R0, (-6)10      # If R10 is not equal to zero, branch to location 400100

```

b) There is a loop which is executed “R10” times or $(100)_{10}$ times. The loop contains 6 instructions. Therefore, the number of instructions executed is 600.

We can calculate the number of clock periods for the program then :

$$\begin{aligned}
 \text{Number of clock periods} &= \sum_{i=1}^6 CPI_i \times I_i = (100 \times 5) + (100 \times 4) + (100 \times 4) + (100 \times 4) + (100 \times 4) + (100 \times 3) \\
 &= 2,400
 \end{aligned}$$

The time to execute the code is the number of clock periods for the code multiplied by the clock period duration :

$$\text{clock period} = \frac{1}{200 \times 10^6} = 5 \times 10^{-9} = 5\text{ns}$$

Time to execute the code = The number of clock periods for the code * The clock period duration

$$= 2400 \times 5 \times 10^{-9} = 12000 \times 10^{-9} = 12 \times 10^3 \times 10^{-9} = 12 \times 10^{-6} = 12\mu\text{seconds}$$

ii)
$$\text{CPI}_{\text{ave}} = \frac{\text{number of clock periods}}{\text{NI}} = \frac{2400}{600} = 4$$

$$\text{MIPS}_{\text{ave}} = \frac{\text{NI}}{\text{CPUtime} \times 10^6} = \frac{600}{12 \times 10^{-6} \times 10^6} = \frac{600}{12} = 50$$

Q6) The clock frequency is 100 MHz. A program is run on EMY. The following table contains the statistics :

Instruction	Number of times run	CPI _i
LW	2000	0, 1, 2, 3, 4 => 5
SW	500	0, 1, 2, 5 => 4
ADD	12000	0, 1, 16, 17 => 4
SUB	4500	0, 1, 2, 5 => 4
BEQ	140	0, 1, 8 => 3
J	50	0, 1, 18 => 3

Calculate :

- Number of clock periods the program takes,
- The CPU time,
- The CPI_{ave} ,
- The MIPS_{ave} .

A6) a)

$$\begin{aligned} \text{number of clock periods} &= \sum_{i=1}^6 \text{NI}_i \text{CPI}_i = (2000 \times 5) + (500 \times 4) + (12000 \times 4) + (4500 \times 4) + (140 \times 3) + (50 \times 3) \\ &= 78,570 \text{ clock periods} \end{aligned}$$

b)

$$\text{clock period} = \frac{1}{100 \times 10^6} = 10 \times 10^{-9} = 10 \text{ ns}$$

$$\text{CPUtime} = \text{number of clock periods} \times \text{clock period} = 78570 \times 10 = 785700 \text{ ns} = 785.7 \mu\text{s}$$

c)

$$\text{NI} = \sum_{i=1}^6 \text{NI}_i = 2000 + 500 + 12000 + 4500 + 140 + 50 = 19,190$$

$$\text{CPI}_{\text{ave}} = \frac{\text{number of clock periods}}{\text{NI}} = \frac{78570}{19190} = 4.09$$

d)

$$MIPS_{ave} = \frac{NI}{CPUtime \times 10^6} = \frac{19190}{785.7 \times 10^{-6} \times 10^6} = \frac{19190}{785.7} = 24.42$$

Q7) The following piece of program is run on the EMY computer :

```

400500 I1      # Instruction I1 is executed 5000 times. Its CPIi is asked below
400504 I2      # Number of times instruction I2 is executed is asked below. The CPIi is 4.
400508 I3      # Instruction I3 is executed 5000 times. The CPIi is 3
40050C I4      # Instruction I4 is executed once. The CPIi is 3
    
```

The clock period is 100MHz. You are given the following exact measurements :

```

NI = 15001
CPU time = 0.55003 * 10-3 seconds
    
```

Calculate :

- i) the number of times I2 is executed,
- ii) the number of clock periods the program takes,
- iii) the CPI_i of I1,
- iv) the CPI_{ave},
- v) the MIPS_{ave}.

A7) i) $NI = 15001 = NI_1 + NI_2 + NI_3 + NI_4 = 5000 + NI_2 + 5000 + 1$

$$NI_2 = 15001 - 10001 = 5000$$

ii)

$$\text{clock period} = \frac{1}{100 \times 10^6} = 10 \times 10^{-9} = 10\text{ns}$$

$$0.55003 \times 10^{-3} = \text{number of clock periods} \times 10^{-8}$$

$$CPUtime = \text{number of clock periods} \times \text{clock period} \quad \left| \quad \text{number of clock periods} = \frac{0.55003 \times 10^{-3}}{10^{-8}} = 0.55003 \times 10^5 = 55003 \right.$$

iii) number of clock periods = 55003 = 5000 * CPI_{I1} + 5000 * 4 + 5000 * 3 + 1 * 3

$$CPI_{I1} = [55003 - (20000 + 15000 + 3)] / 5000 = 20000 / 5000 = 4$$

iv)

$$CPI_{ave} = \frac{\text{number of clock periods}}{NI} = \frac{55003}{15001} = 3.67$$

v)

$$MIPS_{ave} = \frac{NI}{CPUtime \times 10^6} = \frac{15001}{550.03 \times 10^{-6} \times 10^6} = \frac{15001}{550.03} = 27.27$$

Q8) A program is run on the EMY computer and the following set of statistics is obtained :

The total number of memory accesses for the program : 150 million
The number of memory accesses for data : 50 million
The fraction of Load instructions run 0.4
The fraction of integer A/L instructions run : 0.4
The MFLOPS_{ave} = 0.0
The clock frequency = 500 MHz

Obtain as many figures as you can from the given set of statistics.

A8) The total number of memory accesses for the program is the sum of the number of memory accesses for instructions and the number of memory accesses for data. The number of memory accesses for instructions is the number of instructions run for the program (NI) :

$$150M = NI + 50M \Rightarrow NI = 100M$$

$$\text{clock period} = \frac{1}{\text{clock frequency}} = \frac{1}{500 \times 10^6} = 2 \times 10^{-9} \text{seconds} = 2\text{ns}$$

The number of L/S instructions run = the number of memory accesses for data = 50M

The number of Load instructions run = 0.4*NI = 0.4*100M = 40M

The number of Store instructions run = 50M - 40M = 10M

The number of A/L instructions run = 0.4*NI = 0.4*100M = 40M

The number of FP instructions run = 0 since MFLOP_{ave} = 0.0

The number of control instructions run = NI - #A/L - #L/S = 100M - 40M - 50M = 10M

The number of clock periods for the program = #A/L*CPI_{A/L} + #L*S*CPI_L + #S*CPI_S + #control*CPI_{control}

$$= 40M*4 + 40M*5 + 10M*4 + 10M*3 = 160M + 200M + 40M + 30M = 430M$$

$$\text{CPUtime} = \# \text{clock periods for program} \times \text{clock period} = 430 \times 10^6 \times 2 \times 10^{-9} = 0.860\text{sec}$$

$$\text{CPI}_{\text{ave}} = \frac{\# \text{clock periods for program}}{\text{NI}} = \frac{430 \times 10^6}{100 \times 10^6} = 4.3$$

$$\text{MIPS}_{\text{ave}} = \frac{\text{NI}}{\text{CPUtime} \times 10^6} = \frac{100 \times 10^6}{0.860 \times 10^6} = 116.28$$

Q9) A benchmark suit is run on EMY and the following execution time statistics is obtained :

A/L instruction frequency : 35% with $CPI_i = 4.4$

data transfer instruction frequency : 25% with $CPI_i = 4.8$

control instruction frequency : 15% with $CPI_i = 3.3$

FP instruction frequency : 25% with $CPI_i = 7.2$

Time to execute all FP instructions : 6 seconds

clock frequency = 200MHz

Calculate : CPI_{ave} , CPUtime, $MIPS_{ave}$, $MFLOPS_{ave}$, total number of memory accesses made, the number of memory accesses for data only.

A9)

$$CPI_{ave} = \sum_{i=1}^4 CPI_i \times I_i = (4.4 \times 0.35) + (4.8 \times 0.25) + (3.3 \times 0.15) + (7.2 \times 0.25) = 5.035$$

$$\begin{aligned} \# \text{clock periods for FP instructions} &= \text{time for FP instructions} \times \text{clock frequency} \\ &= 6 \times 200 \times 10^6 \\ &= 1.2 \times 10^9 \end{aligned}$$

$$\# \text{of FP operations} = \frac{\# \text{clock periods for FP instructions}}{CPI_{FP}} = \frac{1.2 \times 10^9}{7.2} = 166.7 \times 10^6$$

It is given that 25% of all instructions run are FP instructions and each is a FP operation. Thus, 166.7×10^6 FP instructions are run. Then,

$$NI = \frac{100 \times 166.7 \times 10^6}{25} = 667 \times 10^6$$

$$CPUtime = \frac{NI \times CPI_{ave}}{\text{clock frequency}} = \frac{667 \times 10^6 \times 5.035}{200 \times 10^6} = 16.79 \text{ seconds}$$

$$MIPS_{ave} = \frac{NI}{CPUtime \times 10^6} = \frac{667 \times 10^6}{16.79 \times 10^6} = 39.72$$

$$MFLOPS_{ave} = \frac{\# \text{FP operations}}{CPUtime \times 10^6} = \frac{166.7 \times 10^6}{16.79 \times 10^6} = 9.93$$

The percentage of data transfer instructions that generate memory data accesses is 25. Then,

$$\# \text{data transfer instructions} = \frac{25 \times 667 \times 10^6}{100} = 166.7 \times 10^6$$

A data transfer instruction generates only one data access on the memory. Therefore, there are 166.7×10^6 memory accesses for data.

Finally, the total number of memory accesses is the sum of the memory accesses for instruction fetches and the memory data accesses. Since there are 667×10^6 instructions executed, total number of memory accesses = $667 \times 10^6 + 166.7 \times 10^6 = 833.7 \times 10^6$

Q10) Assume that a program is run on EMY. It is observed that for this program 10 billion (10^{10}) instructions are executed and $\text{CPI}_{\text{ave}} = 6.37$. 30% of all instructions run are Load and Store, 40% of all instructions run are single-precision floating-point and 30% of all instructions run are Integer A/L instructions. The clock frequency is 3 GHz.

Calculate the following figures :

- i) CPUtime
- ii) MIPS_{ave}
- iii) $\text{MFLOPS}_{\text{ave}}$
- iv) Total number of **all** memory accesses made for the program

A10) i) The CPUtime : First, we have to obtain the clock period :

$$\text{clock period} = \frac{1}{3 \times 10^9} = 0.33 \times 10^{-9} = \mathbf{1 \text{ ns}}$$

$$\begin{aligned} \text{CPUtime} &= \text{NI} \times \text{CPI}_{\text{ave}} \times \text{clock period duration} \\ &= 10^{10} \times 6.37 \times 0.33 \times 10^{-9} \text{ seconds} = \mathbf{21.23 \text{ seconds}} \end{aligned}$$

ii) The MIPS_{ave} for the program :

$$\text{MIPS}_{\text{ave}} = \frac{\text{NI}}{\text{CPUtime} \times 10^6} = \frac{10^{10}}{21.23 \times 10^{-9} \times 10^6} = \mathbf{471.03}$$

iii) To calculate the $\text{MFLOPS}_{\text{ave}}$, we need to calculate the number of floating-point operations for the program. It is given that 40% of all instructions are FP. Then, the number of FP instructions run is $= 4 \times 10^{10} = 4 \times 10^9$. The $\text{MFLOPS}_{\text{ave}}$ for the program :

$$\text{MFLOPS}_{\text{ave}} = \frac{\text{Number of FP operations}}{\text{CPUtime} \times 10^6} = \frac{4 \times 10^9}{21.23 \times 10^{-9} \times 10^6} = \mathbf{188.41}$$

iv) We know that every instruction requires a memory access to fetch the instruction. In addition, Load and Store instructions require a memory access for data. It is given that 30% of instructions executed are Load and Store instructions. Then, the total number of memory accesses is calculated by adding the number of instruction fetches and the number of Load and Store instructions requiring data accesses : $10^{10} + 0.3 \times 10^{10} = 1.3 \times 10^{10}$.

Q11) Consider the following piece of EMY mnemonic machine language program :

```

400000    LW      R10, 0(R8)          # R8 points at array A and initially has 10000000
400004    LW      R11, 0(R9)          # R9 points at array B and initially has 10002000
400008    SLT    R12, R10, R11        # Compare R10 and R11
40000C    BEQ    R12, R0, 1           # Skip next instruction if R10 is not less than R11
400010    SW      R11, 0(R8)          # Store the 2nd number in the first
400014    ADDI   R8, R8, 4             # Update the array A pointer
400018    ADDI   R9, R9, 4             # Update the array B pointer
40001C    ADDI   R13, R13, (-1)10    # The loop-end counter, R13, has 2 initially
400020    BNE    R13, R0, (-9)10     # If not the end, go back to 400000
-----
10000000  2
10000004  9
-----
10002000  3
10002004  6

```

i) If the clock frequency is **1 GHz**, determine how long it takes to run the above piece of program as done **in class** ? Note that each memory access takes **one** clock period.

ii) Assume that a **new** machine language instruction is created to perform the above program faster, SLTM :

SLTM Rd, (Rs), (Rt) # If M[Rs] < M[Rt] then Rd ← 1, else Rd ← 0

CPI_{SLTM} is **6** since we trace states 0, 1, 16, 17, 18 and 19

The above program is rewritten by using the **SLTM** instruction as follows :

```

400000    SLTM   R12, (R8), (R9)      # Compare array A and B elements
400004    BEQ    R12, R0, 2           # Skip next instruction if array A element is not less than array B element
400008    LW      R11, 0(R9)          # Read array B element
40000C    SW      R11, 0(R8)          # Store array B element in array A
400010    ADDI   R8, R8, 4             # Update the array A pointer
400014    ADDI   R9, R9, 4             # Update the array B pointer
400018    ADDI   R13, R13, (-1)10    # The loop-end counter, R13, has 2 initially
40001C    BNE    R13, R0, (-8)10     # If not the end, go back to 400000

```

The SLTM instruction replaces the first **three** instructions.

Determine the **new** execution time of the program as done **in class**. Again, assume that the clock frequency is **1 GHz** and each memory access takes **one** clock period.

A11) i) The loop compares arrays A and B. If an array A element is less than the corresponding array B element, the array A element is replaced with the array B element by executing a SW. Out of two comparisons (iterations), only one requires the execution of the SW instruction. Therefore, the following instructions are run : LW, LW, SLT, BEQ, SW, ADDI, ADDI, ADDI, BNE, LW, LW, SLT, BEQ, ADDI, ADDI, ADDI, BNE.

The execution timings of the six different instructions in the loop are as follows :

LW : 0, 1, 2, 3, 4 => 5 cp BEQ : 0, 1, 8 => 3 cp ADDI : 0, 1, 2, 16 => 4 cp
SLT : 0, 1, 6, 7 => 4 cp SW : 0, 1, 2, 5 => 4 cp BNE : 0, 1, 16 => 3 cp

The number of clock periods to run the 17 instructions is 5 + 5 + 4 + 3 + 4 + 4 + 4 + 4 + 3 + 5 + 5 + 4 + 3 + 4 + 4 + 4 + 3 = 68 clock periods.

$$\text{clock period} = \frac{1}{1 \times 10^9} = 1 \times 10^{-9} \text{ sec} = \mathbf{1ns} \quad \text{The execution time is } 68 * 1 = 68\text{ns}$$

ii) We execute the following 14 instructions : SLTM, BEQ, LW, SW, ADDI, ADDI, ADDI, BNE, SLTM, BEQ, ADDI, ADDI, ADDI, BNE.

The number of clock periods to run the 14 instructions is $6 + 3 + 5 + 4 + 4 + 4 + 4 + 3 + 6 + 3 + 4 + 4 + 4 + 3 = 57$ clock periods. Then, the execution time is $57 * 1 = 57\text{ns}$. The speedup is $68/57 = 1.19$ or 19%

Compared with the old code, the new code has one less instruction, **not** two. An **extra** LW instruction is needed to bring the array B element to the CPU to write to array A even if the SLTM instruction brings the same element to the CPU. This is because the SLTM instruction does **not** keep the element in an architectural register that can be used by other instructions.

In general, if data read from the memory is used again, it must be kept on architectural registers as long as possible. The RISC concept of having only LW and SW to access the memory for data supports that idea. CISC A/L instructions accessing memory for data keep the data in organizational registers, not visible to the machine language programmer. It means if the same data element is needed an additional instruction is needed !

Q12) Assume that a program is run on EMY. The following table contains the statistics of the execution :

Instruction	Number of times run	CPI _i
LW	1M	5
SW	0.25M	4
ADD	2M	4
BEQ	0.25M	3

Instruction	Number of times run	CPI _i
FPADD	10M	7
FPMUL	10M	11
FPDIV	10M	20

The clock frequency is 2 GHz.

a) Calculate the following figures :

- i) The CPUtime
- ii) The CPI_{ave}
- iii) The MIPS_{ave}
- iv) The MFLOPS_{ave}

b) Assume that we decide to improve the execution of FPDIV instructions : we execute a FPDIV instruction in **10** clock periods.

Calculate the Speedup_{overall} figure.

A12) clock period = $\frac{1}{2 \times 10^9} = 0.5 \times 10^{-9} = \mathbf{0.5\text{ns}}$

a) i) First, we need to get the number of clock periods the program takes :

$$\text{number of clock periods for the program} = \sum_{i=1}^7 I_i \text{CPI}_I$$

$$= (1\text{M} \times 5) + (0.25\text{M} \times 4) + (2\text{M} \times 4) + (0.25 \times 3) + (10\text{M} \times 7) + (10\text{M} \times 11) + (10\text{M} \times 20) = \mathbf{394.75 \times 10^6}$$

The CPU time can now be calculated :

$$\begin{aligned} \text{CPUtime} &= \text{clock periods for the program} \times \text{clock period duration} \\ &= 394.75 \times 10^6 \times 0.5 \times 10^{-9} = \mathbf{0.197375 \text{ seconds}} \end{aligned}$$

ii) The CPI_{ave} :

$$\begin{aligned} \text{NI} &= \sum_{i=1}^7 I_i = 1\text{M} + 0.25\text{M} + 2\text{M} + 0.25\text{M} + 10\text{M} + 10\text{M} + 10\text{M} = \mathbf{33.5 \times 10^6} \\ \text{CPI}_{\text{ave}} &= \frac{\text{number of clock periods for the program}}{\text{NI}} = \frac{394.75 \times 10^6}{33.5 \times 10^6} = \mathbf{11.78} \end{aligned}$$

iii) The MIPS_{ave} :

$$\text{MIPS}_{\text{ave}} = \frac{\text{NI}}{\text{CPUtime} \times 10^6} = \frac{33.5 \times 10^6}{0.197375 \times 10^6} = \mathbf{169.73}$$

iv) The $\text{MFLOPS}_{\text{ave}}$:

$$\text{Number of FP instructions} = \text{NI}_{\text{FPADD}} + \text{NI}_{\text{FPMUL}} + \text{NI}_{\text{FPDIV}} = 10\text{M} + 10\text{M} + 10\text{M} = \mathbf{30 \times 10^6}$$

$$\text{MFLOPS}_{\text{ave}} = \frac{\text{Number of FP operations}}{\text{CPUtime} \times 10^6} = \frac{30 \times 10^6}{0.197375 \times 10^6} = \mathbf{151.99}$$

b) The new number of clock periods for the programs is as follows :

$$\begin{aligned} \text{number of clock periods for the program} &= \sum_{i=1}^7 I_i \text{CPI}_i \\ &= (1\text{M} \times 5) + (0.25\text{M} \times 4) + (2\text{M} \times 4) + 0.25 \times 43 + (10\text{M} \times 7) + (10\text{M} \times 11) + (10\text{M} \times 10) = \mathbf{294.75 \times 10^6} \end{aligned}$$

The new CPUtime is as follows :

$$\begin{aligned} \text{CPUtime} &= \text{Number of clock periods for the program} \times \text{clock period} \\ &= 294.75 \times 10^6 \times 0.5 \times 10^{-9} = \mathbf{0.147375 \text{ seconds}} \end{aligned}$$

$$\text{Speedup}_{\text{overall}} = \frac{\text{CPUtime}_{\text{old}}}{\text{CPUtime}_{\text{new}}} = \frac{0.197375}{0.147375} = \mathbf{1.34}$$

Q13) Consider the following piece of EMY mnemonic machine language program :

```

400000   ADDI   R8, R0, 0
400004   LW    R9, 0(R10)           # R10 points at array A and initially has 1000000
400008   ADD   R8, R8, R9
40000C   ADDI   R10, R10, 4
400010   ADDI   R11, R11, (-1)10   # R11 is the loop-end counter
400014   BNE   R11, R0, (-5)10
400018   SW    R8, 0(R10)

```

Assume that the EMY CPU is **unpipelined**, as discussed in **Chapter 5** of the textbook and there is a **perfect memory**, with no stalls. Assume also that R11 initially has **(1000)₁₀**.

The instruction execution timings for the above program are as follows : $CPI_{LW} = 5$, $CPI_{ADD} = 4$, $CPI_{ADDI} = 4$, $CPI_{BNE} = 3$ and $CPI_{SW} = 4$. Assume also that the clock frequency is 2 GHz.

Calculate the following figures :

- i) CPI_{ave}
- ii) CPUtime
- iii) $MIPS_{ave}$
- iv) The total number of **all** memory accesses made for the program

A13) We execute the following instructions for “k” iterations : $ADDI + k(LW + ADD + ADDI + ADDI + BNE) + SW$

i) “k” is given as $(1000)_{10}$. Then, the number of clock periods for the program based on the given CPI figures is : $4 + 1000(5 + 4 + 4 + 4 + 3) + 4 = 20008$.

The number of instructions executes is : $1 + 1000(1 + 1 + 1 + 1 + 1) + 1 = 5002$

$$CPI_{ave} = \frac{\text{Number of clock periods for the program}}{NI} = \frac{20008}{5002} = 4$$

ii) The CPUtime : First, we have to obtain the clock period :

$$\text{clock period} = \frac{1}{2 \times 10^9} = 0.5 \times 10^{-9} = \mathbf{0.5ns}$$

$$\begin{aligned} \text{CPUtime} &= NI \times CPI_{ave} \times \text{clock period duration} \\ &= 5002 \times 4 \times 0.5 \times 10^{-9} \text{seconds} = 10004 \times 10^{-9} \text{seconds} = \mathbf{10.004\mu\text{seconds}} \end{aligned}$$

iii) The $MIPS_{ave}$ for the program :

$$MIPS_{ave} = \frac{NI}{\text{CPUtime} \times 10^6} = \frac{5002}{10.004 \times 10^{-6} \times 10^6} = \mathbf{500}$$

iv) We know that for every instruction a memory access is performed. In addition, Load and Store instructions require a memory access for data. Then, the total number of memory accesses is calculated by adding the number of instruction fetches and the number of data accesses due to Load and Store instructions : $1 + 1000(2 + 1 + 1 + 1 + 1) + 2 = 6003$.

Q14) Consider the following piece of EMY mnemonic machine language program :

```

400C00    LW      R9, 0(R8)           # R8 points at array A and initially has 10EA0000
400C04    ADD     R9, R9, R9
400C08    ADDI    R10, R9, AC
400C0C    SW      R10, 0(R8)
400C10    ADDI    R8, R8, 4
400C14    ADDI    R11, R11, (-1)10   # R11 is the loop-end counter
400C18    BNE    R11, R0, (-7)10

```

Assume that the EMY CPU is **unpipelined**, as discussed in **Chapter 5** of the textbook and there is a **perfect memory**, with no stalls. Assume also that R11 initially has **(10,000)₁₀**.

The instruction execution timings for the above program are as follows : $CPI_{LW} = 5$, $CPI_{ADD} = 4$, $CPI_{ADDI} = 4$, $CPI_{SW} = 4$ and $CPI_{BNE} = 3$. Assume also that the clock frequency is 2 GHz.

Calculate the following figures :

- i) CPI_{ave}
- ii) CPUtime
- iii) $GIPS_{ave}$
- iv) The total number of **all** memory accesses made for the program.

A14) We execute the following instructions for “k” iterations : (LW + ADD + ADDI + SW + ADDI + ADDI + BNE)

i) “k” is given as $(10,000)_{10}$. Then, the number of clock periods for the program based on the given CPI figures is : $10000(5 + 4 + 4 + 4 + 4 + 4 + 3) = 280000$.

The number of instructions executed is : $10000(1 + 1 + 1 + 1 + 1 + 1) = 70000$

$$CPI_{ave} = \frac{\text{Number of clock periods for the program}}{NI} = \frac{280000}{70000} = 4$$

ii) The CPUtime : First, we have to obtain the clock period :

$$\text{clock period} = \frac{1}{2 \times 10^9} = 0.5 \times 10^{-9} = \mathbf{0.5ns}$$

$$\begin{aligned} \text{CPUtime} &= NI \times CPI_{ave} \times \text{clock period duration} \\ &= 70000 \times 4 \times 0.5 \times 10^{-9} \text{ seconds} = 140000 \times 10^{-9} \text{ seconds} = \mathbf{140\mu\text{seconds}} \end{aligned}$$

iii) The $GIPS_{ave}$ for the program :

$$GIPS_{ave} = \frac{NI}{\text{CPUtime} \times 10^9} = \frac{70000}{140000 \times 10^{-9} \times 10^9} = 0.5$$

iv) We know that for every instruction a memory access is performed. In addition, Load and Store instructions require a memory access for data. Then, the total number of memory accesses is calculated by adding the number of instruction fetches and the number of data accesses due to Load and Store instructions : $10000(2 + 1 + 1 + 2 + 1 + 1 + 1) = \mathbf{90000}$.