DUE: TBA

READ:

i) Related portions of Chapter 4 (except Sections 4.4 through 4.10)
ii) Related portions of Appendix A
iii) Related portions of Appendix B
iv) Related portions of Appendix D

ASSIGNMENT: There are five questions.

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

1) Consider Question 3 of Homework 3 where the EMY CPU high-level state diagram and datapath are modified for the JR instruction.

Based on the feedback for your Homework 3 solution, modify the low-level state diagram. If you decide to add new states, start at state 16. Then, modify the Control Unit of the EMY CPU for JR. Assume that the EMY CPU is hardwired and already executes those nine instructions in the EMY CPU handout.

In order to solve the problem, first show the high-level state diagram obtained in Homework 3 and then modify the low-level state diagram and the Control Unit.

2) Consider Question 4 of Homework 3 where the EMY CPU high-level state diagram, datapath and low-level state diagram are modified for the JAL instruction.

Based on the feedback for your Homework 3 solution, modify the low-level state diagram. If you decide to add new states, start at state 16. Then, modify the Control Unit of the EMY CPU for JAL. Assume that the EMY CPU is hardwired and already executes those nine instructions in the EMY CPU handout.

In order to solve the problem, first show the high-level state diagram obtained in Homework 3 and then modify the low-level state diagram and the Control Unit.

3) Consider the EMY instruction SLTI. Modify the EMY CPU completely to run the instruction.
In order to solve this question, you will assume the CPU is a multicycle CPU. You will use the EMY
CPU handout, by xeroxing it and modifying the necessary pages of the xeroxed copy. If you do not
want to copy and modify the handout, you can just show the changes to the datapath as done in
past exam questions below.

In this question you will modify the EMY CPU so that it can execute the SLTI instruction. You
need to modify the high-level state diagram (not in terms of buses) in parallel with the modifica-
tion of the datapath. Then, you will modify the low-level state diagram. If you decide to add
new states, start at state 16.

Then, you will modify the Control Unit of the EMY CPU for SLTI. Assume that the EMY CPU
is hardwired and already executes those nine instructions in the EMY CPU handout.

4) Consider the first program given on the first page of the EMY Mnemonic Machine Language
Programming Examples Handout. It takes the EMY CPU 18 clock periods to run these four
instructions. If the clock frequency is 1GHz, it would take the CPU 18ns to run the instructions.

Not satisfied with this, we decide to have clock quadrupling where the CPU clock frequency is
now at 4 GHz (the clock period is 0.25ns), while the memory still takes 1ns per access. How long
does it take to run these four instructions now?

5) Consider the fourth program given on the fifth page of the EMY Mnemonic Machine Lan-
guage Programming Examples Handout. It is a function with seven instructions. It multiplies
two numbers. Assume that the numbers multiplied are $Y = 2$ and $Z = 6$. Assume also that the JR
instruction takes 3 clock periods to run since we trace states 0, 1 and 16.

i) If the clock frequency is 1GHz, how long will it take to run the function?

ii) Not satisfied with this, we decide to have clock doubling where the CPU clock frequency is
now at 2GHz (the clock period is 0.5ns), while the memory still takes 1ns per access. How long
does it take to run the function now?

**RELEVANT QUESTIONS AND ANSWERS**

**Q1** We have decided to modify the data and control units of the CPU in Handout 11 for a new instruction. Its syn-
tax and architectural operation are as follows:

\[
\text{MTRADD} \quad \text{Rd, (Rs), (Rt)} \quad \text{Rd} \leftarrow M[\text{Rs}] + M[\text{Rt}]
\]

The brief description of the new instruction called, Memory-To-Register Add, is that it adds two memory locations
pointed by “Rs” and “Rt” and stores the result in register “Rd.”
i) Show the modified portion of the high-level state diagram (not in terms of buses). How long does it take to run the instruction? Later, when we cover Chapter 4, we will call it CPI of the instruction. Then, what is the CPI of the instruction?

ii) Show the modified portion of the data unit together with control signals. Note that this is a CISC instruction since an A/L instruction accesses memory for data.

A1) i) There are multiple solutions. One which is not too slow and not too expensive is given here. The modified high-level state diagram is as follows:

The CPI is seven (7) since we trace states 0, 1, 16, 17, 18, 19 and 20.

ii) The modified portion of the data unit is as follows:
Q2) Consider the following piece of EMY mnemonic machine language program:

```
400000 ADD R8, R9, R0
400004 SLT R10, R9, R0
400008 BEQ R10, R0, 1
40000C SUB R8, R0, R9
```

The above piece of code is completely implemented by the following instruction:

```
ABS   Rd, Rs  # Rd ← |Rs|
```

i) Rewrite the above code by using the new instruction which takes the absolute value of Rs.

ii) Assume that the high-level state diagram is modified to run the new machine language instruction as follows:

![State Diagram]

Also, the EMY CPU datapath is modified as follows:

![Datapath Diagram]

Obtain the low-level state diagram based on the modifications above.
iii) Assume that we run the following instruction:

\[
400000 \quad \text{ABS} \quad \text{R14, R15}
\]

Show the values of registers and control signals for five clock periods for which continue with the following table:

<table>
<thead>
<tr>
<th>Cp</th>
<th>State</th>
<th>PC</th>
<th>IR</th>
<th>ALUSrcA</th>
<th>RegWriteCond</th>
<th>ALUop</th>
<th>A</th>
<th>B</th>
<th>ALUout</th>
<th>R14</th>
<th>R15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>NS</td>
<td>NS</td>
<td>00</td>
<td>0</td>
<td>00</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>NS</td>
<td>NS</td>
<td>...</td>
</tr>
</tbody>
</table>

A2) i) The new code is as follows:

\[
400000 \quad \text{ABS} \quad \text{R8, R9} \quad # \quad \text{R8} \quad \rightarrow \quad | \text{R9} |
\]

ii) The modified EMY low-level state diagram is as follows:
iii) The table with the values is as follows:

<table>
<thead>
<tr>
<th>Cp</th>
<th>State</th>
<th>PC</th>
<th>IR</th>
<th>ALUSrcA</th>
<th>RegWriteCond</th>
<th>ALUop</th>
<th>A</th>
<th>B</th>
<th>ALUout</th>
<th>R14</th>
<th>R15</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initial</td>
<td>---</td>
<td>400000</td>
<td>?</td>
<td>---</td>
<td>---</td>
<td>?</td>
<td>?</td>
<td>(-6)10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>NS</td>
<td>NS</td>
<td>00</td>
<td>0</td>
<td>00</td>
<td>?</td>
<td>?</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>2</td>
<td>400004</td>
<td>ABS</td>
<td>R14, R15</td>
<td>00</td>
<td>0</td>
<td>00</td>
<td>?</td>
<td>?</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>NS</td>
<td>NS</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>(-6)10</td>
<td>0</td>
<td>?</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>NS</td>
<td>NS</td>
<td>01</td>
<td>0</td>
<td>10</td>
<td>(-6)10</td>
<td>0</td>
<td>FFFFFFFA</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>NS</td>
<td>NS</td>
<td>10</td>
<td>1</td>
<td>01</td>
<td>(-6)10</td>
<td>0</td>
<td>1</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>NS</td>
<td>NS</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>(-6)10</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>NS</td>
</tr>
</tbody>
</table>

Q3) Assume that the EMY CPU datapath is modified as follows:

i) Assume that the above modification in the datapath is for a new machine language instruction. The corresponding low-level state diagram is below.
Obtain the corresponding **high-level** state diagram. How many clock periods does it take to run the new instruction?

**ii)** The new datapath allows **new** microoperations. **List** at least **four** (4) new microoperations that are **not** shown in the new states.

**iii)** Describe the **syntax**, **semantics**, **format**, etc. of the **new** machine language instruction. If a **new** addressing mode or syntax is encountered, indicate so.

**A3)**

**i)** The modified EMY high-level state diagram (not in terms of buses) is as follows:

```
<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>same</td>
</tr>
<tr>
<td>1</td>
<td>same</td>
</tr>
<tr>
<td>2</td>
<td>LW, SW, ?</td>
</tr>
<tr>
<td>3</td>
<td>same</td>
</tr>
<tr>
<td>16</td>
<td>MDR</td>
</tr>
<tr>
<td>17</td>
<td>ALUout</td>
</tr>
<tr>
<td>18</td>
<td>GPR[Rt]</td>
</tr>
</tbody>
</table>
```

**ii)** The **new** microoperations not listed in the above high-level state diagram include the following:

- M[MDR] ← B
- ALUout ← MDR + B
- ALUout ← MDR + 4
- ALUout ← MDR + DOIimm+
- ALUout ← MDR + (DOIimm+ * 4)
- ALUout ← MDR op B
- ALUout ← MDR op 4
- ALUout ← MDR op DOIimm+
- ALUout ← MDR op (DOIimm+ * 4)
- PC ← MDR + B
- PC ← MDR + DOIimm+
- PC ← MDR + (DOIimm+ * 4)
- PC ← MDR op B
- ... more ...

**iii)** The new instruction adds a memory location (pointed by another memory location) and a register. This instruction can be called **ADDMIR** : Add memory indirect register:

- **The syntax** of the new instruction: ADDMIR Rt, (Disp(Rs))
- **The semantics** of the new instruction: Rt ← M[M[Rs + Disp^*]] + Rt
- **The format** is the **I format**:

```
<table>
<thead>
<tr>
<th>Opcode</th>
<th>Rs</th>
<th>Rt</th>
<th>DOIimm</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>5</td>
<td>5</td>
<td>16</td>
</tr>
</tbody>
</table>
```

**Three** arguments are used by the instruction: The second source argument is a register argument: Rt. We use the register addressing mode for it. The destination is implied to be Rt again and so the Implied addressing mode is used.
for it. The first source argument is a memory argument. It is pointed by another memory location which is a new addressing mode. We use the Memory Indirect via 2-byte signed displacement addressing mode for it.

**Q4** The EMY high-level state diagram has been modified as follows:

![Diagram](image)

a) Draw the modified portion of the EMY CPU datapath.

b) Show the corresponding low-level state diagram.

**A4** a) We see that the ALU has to add B which always has Rt and 4. That is ABUS and BBUS are added such that they carry values B (Rt) and 4. In the datapath, already number 4 is connected to the BBUS MUX which is MUX5. Thus, we only need to connect register B to the ABUS MUX which is MUX4. Now, MUX4 has to be larger and receive two ALUSrcA control signals.

One could connect a 4 to MUX4 for the addition. However, connecting Register B to MUX4 can handle upgrades better since it is quite possible that B and DOImm* and DOImm*<<2 might have to be operated on in the future.

In addition, the result of the ALU is directly stored on Rt, bypassing the ALUout register. Then, we connect the output of the ALU directly to the WBUS MUX which is MUX3. MUX3 has to be larger and receive two MemtoReg control signals. The modified datapath is then as follows:

![Diagram](image)

b) The corresponding low-level state:

```
ALUSrcA = 10
ALUSrcB = 01
ALUop = 00
RegDst = 0
MemtoReg = 10
RegWrite = 1
```
Q5) A new machine language instruction is added to the EMY architecture. The following table is obtained by observing the CPU when it runs this new instruction:

<table>
<thead>
<tr>
<th>clock period</th>
<th>State</th>
<th>PC</th>
<th>IR</th>
<th>R9</th>
<th>R12</th>
<th>M[4000F0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>---</td>
<td>4000F0</td>
<td>?</td>
<td>3F</td>
<td>?</td>
<td>YDI</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4000F4</td>
<td>YDI</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>FFFFFFFC0</td>
<td>NS</td>
</tr>
</tbody>
</table>

The states mentioned above are the EMY high-level state diagram states. The letters “YDI” mean “You determine it.”

a) What is the CPI of this new instruction?

b) Describe the new instruction architecturally, i.e. its instruction format, what it accomplishes,... Also, indicate what “YDI” is on the table mnemonically (not in HEX).

c) Modify the EMY CPU data unit to be able to run this new instruction.

d) Show the modified low-level state diagram.

e) Assume that the Control Unit is hardwired. Based on the table above and your answer to parts (c) and (d), modify the hardwired EMY Control unit. That is, show the modified portion of the hardware of the control unit.

A5) a) The CPI of the instruction is 4 since we trace the states 0, 1, 6 and 7.

b) State 6 is taken after state 1, therefore, it is an R-type instruction. We see that only two GPR registers are used for data: R9 and R12. Also, we see that one must be used as a source and the other as a destination. These imply that a unary operation is performed on the source and stored on the destination by the instruction. We then work on the source-destination relationship and so convert both numbers to binary:

```
0000003F                   0000 0000 0000 0000 0000 0011 1111
FFFFFFFC0                 1111 1111 1111 1111 1111 1100 0000
```

The two numbers are the complement of each other. Thus, we have a complement instruction with the R-format:

```
COMP  Rd, Rs  Rd  Rs
```

<table>
<thead>
<tr>
<th>opcode</th>
<th>Rs</th>
<th>Unused</th>
<th>rd</th>
<th>shamt</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

The instruction run in the question (YDI) is COMP R12, R9

c) The only Data Unit change is in the ALU. The ALU must now complement the value on its ABUS input. In order to indicate that it is a complement, an ALUcontrol bit combination must be assigned the complement operation. We select combination 1000 for it:
### ALUop | ALU operation | ALUcontrol
--- | --- | ---
00 | Add | 0010
01 | Sub | 0110
10 | indicated by function field of IR: | 
Add | 0010
Sub | 0110
And | 0000
Or | 0001
SLT | 0111
COMP | 1000

**d)** The low-level state diagram does not need any change since the high-level state diagram is the same, the Data Unit is the same with the exception that the ALU has a new operation to perform and the code to indicate the new operation in the low-level state diagram is the same (ALUop = 10).

**e)** In the hardwired EMY control unit, nothing changes except the ALUcontrol circuit:

The ALUcontrol circuit now outputs a new combination, 1000, if the 2nd opcode (function) indicates it is a COMP instruction.

---

**Q6)** Consider the following instruction that does **not** exist in the EMY instruction set:

\[
\text{SWAPMR} \quad \text{Rt, Disp(Rs)} \quad \# \; \text{Rt} \quad \rightarrow \quad \text{M[Rs + Disp]}^+ \]

**a)** i) Modify the EMY CPU **high-level** state diagram, as done in class.

ii) What is CPI\text{SWAPMR}? Explain.

**b)** Modify the EMY CPU **datapath** to be able to run the new instruction, as done in class.

**c)** Modify the EMY CPU **low-level** state diagram accordingly and as done in class.

**d)** Assume that the EMY CPU control unit is **hardwired**. Assume also that the opcode of the instruction is 17.

Modify the **hardwired** EMY CPU control unit accordingly, as done in class and as follows: Show **two** (2) signals that are modified or new.

Again, just show the circuits for any **two** modified/new signals.

**A6)** **a)** i) The modified **high-level** state diagram of this high-speed implementation is shown below.

ii) CPI\text{SWAPMR} is 6 since we trace states 0, 1, 2, 3, 5 and 4

**b)** The changes in the datapath are that MDR and ALUout are **not** clocked every clock period:
c) The modified EMY low-level state diagram is as follows:

States 4 - 11: Same except ALUoutWrite = 1

d) Two new signals in the control unit, ALUoutWrite and MDRWrite are shown below:

ALUoutWrite is 1 when it is state 1 or 2 or 6
ALUoutWrite = S1 + S2 + S6

MDRWrite is 1 when it is state 3
MDRWrite = S3

Two signals are modified: NS2 and NS0

Other signals modified: MemWrite, IorD, MemtoReg, RegWrite
Q7) 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 R12 is not less than R0
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
```

a) 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 clock doubling is used to speed up the processor. Calculate the new execution time.

b) 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
```

Rewrite the above piece of code, by using the SLTM instruction. Add comments to your program.

c) Modify the EMY high-level state diagram (not in terms of buses) and the datapath to be able to run the new instruction, as done in class. What is CPI_{SLTM}?

d) i) Based on your answers to part (b) and part (c), 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.

ii) Assume that clock doubling is used to speed up the processor. Calculate the new execution time.

A7) a) 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
SLT : 0, 1, 6, 7 => 4 cp
BEQ : 0, 1, 8 => 3 cp
ADDI : 0, 1, 2, 16 => 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 + 3 + 5 + 5 + 4 + 3 + 4 + 4 + 4 + 3 = 68 clock periods. The execution time is 68ns which is computed as follows:

\[
\text{clock period} = \frac{1}{1 \times 10^9} = 1 \times 10^{-9} \text{ sec} = \text{1ns}
\]

The execution time is 68 * 1 = 68ns

ii) If clock doubling is used the instruction timings is as follows:

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

The number of clock periods to run the 17 instructions is 7 + 7 + 5 + 4 + 6 + 5 + 5 + 4 + 7 + 5 + 4 + 5 + 5 + 4 = 90 clock periods. The execution time is 45ns which is computed as follows:

\[
\text{clock period} = \frac{1}{2 \times 10^9} = 0.5 \times 10^{-9} \text{ sec} = 0.5 \text{ns}
\]

The execution time is 90 * 0.5 = 45ns

b) The SLTM instruction replaces the first three instructions:

```
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
```

c) The modified high-level state diagram and the datapath are as follows:

```

```

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

d) i) 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 + 3 + 6 + 3 + 4 + 4 + 4 + 3 = 57 clock periods. Then, the execution time is 57 * 1 = 57ns. 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!
ii) If clock doubling is used the instruction timings is as follows:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Timing</th>
</tr>
</thead>
<tbody>
<tr>
<td>LW</td>
<td>0*, 1, 2, 3*, 4 =&gt; 7 cp</td>
</tr>
<tr>
<td>SLTM</td>
<td>0*, 1, 16*, 17*, 18, 19 =&gt; 9 cp</td>
</tr>
<tr>
<td>BEQ</td>
<td>0*, 1, 8 =&gt; 4 cp</td>
</tr>
<tr>
<td>ADDI</td>
<td>0*, 1, 16, 17 =&gt; 5 cp</td>
</tr>
<tr>
<td>SW</td>
<td>0*, 1, 2, 5* =&gt; 6 cp</td>
</tr>
<tr>
<td>BNE</td>
<td>0*, 1, 16 =&gt; 4 cp</td>
</tr>
</tbody>
</table>

The number of clock periods to run the 14 instructions is $9 + 4 + 7 + 6 + 5 + 5 + 4 + 9 + 4 + 5 + 5 + 5 + 4 = 77$ clock periods. The execution time is 38.5ns which is computed as follows:

$$\text{clock period} = \frac{1}{2 \times 10^9} = 5 \times 10^{-9} \text{sec} = 0.5 \text{ns}$$

The execution time is $77 \times 0.5 = 38.5$ ns.

Q8) 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
400008 ADD R8, R8, R9
40000C ADDI R10, R10, 4  # Update the array A pointer
400010 ADDI R11, R11, (-1)_10  # The loop-end counter, R11, has 2 initially
400014 BNE R11, R0, (-5)  # If not the end, go back
400018 SW R8, 0(R10)
```

i) If the clock frequency is 1 GHz, determine how long it takes to run the above piece of program as done in class.

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

```
ADDMR   Rd, Rs, (Rt)++ # Rd ←← Rs + M[Rt] then Rt ←← Rt + 4
```

Rewrite the above piece of code, by using the ADDMR instruction. Add comments to your program.

iii) Modify the EMY high-level state diagram and the datapath to be able to run the new instruction, as done in class. What is $\text{CPI}_{ADDMR}$?

iv) Based on your answers to part (ii) and part (iii), determine the new execution time of the program as done in class. Again, assume that the clock frequency is 1 GHz.

A8) i) The loop adds elements of array A and stores the result in the location following the array. There are two iterations of the loop indicated by R11 which is initialized to 2. Therefore, the following instructions are run: ADDI + 2(LW + ADD + ADDI + ADDI + BNE) + SW.

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

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Timing</th>
</tr>
</thead>
<tbody>
<tr>
<td>LW</td>
<td>0, 1, 2, 3, 4 =&gt; 5 cp</td>
</tr>
<tr>
<td>BNE</td>
<td>0, 1, 16 =&gt; 3 cp</td>
</tr>
<tr>
<td>ADDI</td>
<td>0, 1, 6, 7 =&gt; 4 cp</td>
</tr>
<tr>
<td>ADDI</td>
<td>0, 1, 16, 17 =&gt; 4 cp</td>
</tr>
<tr>
<td>SW</td>
<td>0, 1, 2, 5 =&gt; 4 cp</td>
</tr>
</tbody>
</table>

The number of clock periods to run the 12 instructions is $4 + 2(5 + 4 + 4 + 4 + 3) + 4 = 48$ clock periods.

```
400000 ADDI R8, R0, 0  # Initialize R8 to 0
400004 ADDMR R8, R8, (R10)++  # Add an array A element pointed by R10 to R8 then update R10
400008 ADDI R11, R11, (-1)_10  # The loop-end counter, R11, has 2 initially
40000C BNE R11, R0, (-3)_10  # If not the end, go back to location 400004
400010 SW R8, 0(R10)  # Store the result of the addition in the location following array A
```
iii) There are a number of different solutions each with different speed and cost. The below implementation is a high speed one. The modified high-level state diagram and datapath of this high-speed implementation are as follows:

\[
\begin{align*}
\text{CPI}_{\text{ADDMR}} & \text{ is 5 since we trace states } 0, 1, 16, 17 \text{ and } 18 \\
\end{align*}
\]

iv) We execute the following eight instructions: \( \text{ADDI} + 2(\text{ADDMR} + \text{ADDI} + \text{BNE}) + \text{SW} \). The number of clock periods to run the eight instructions is \( 4 + 2(5 + 4 + 3) + 4 = 32 \) clock periods. Given that the clock frequency is the same, the execution time is \( 32 \times 1 = 32 \) ns.

Q9) Assume that the EMY CPU low-level state diagram and the datapath are modified to be able to run a new instruction as follows:

\[
\begin{align*}
\text{States 2 - 9 : Same except :} \\
\text{State 5 : Mwbussel = 0} \\
\end{align*}
\]

a) Modify the EMY CPU high-level state diagram accordingly and as done in class. If a state is the same as the original, just write “Same.”

What is the CPI of the new instruction? Explain.

b) What is the new instruction? Indicate only the following: The syntax, semantics, format and the memory accesses made. Determine an unused opcode for the instruction.

c) Assume that the EMY CPU control unit is hardwired.
Modify the **hardwired** EMY CPU **control unit** accordingly, as done in class and as follows:

- Show all control signals that are modified or new, and
- Show one (1) next state signal modified or new.

**A9) a)** The modified high-level state diagram is as follows:

![High-level state diagram](image)

CPI\(_7\) is 4 since we trace states 0, 1, 16 and 17

**b)** On the high level state diagram, we see that we add registers A and B, meaning we add Rs and Rt. Then, we store the sign extended DOImm to a memory location whose address is Rs + Rt. This means we move a **constant** to a memory location whose address is the sum of Rs and Rt: MVMC (Move Memory Constant):

**Syntax:** MVMC (Rs, Rt), Imm

**Semantics:** M[ Rs + Rt] –> Imm**

**Format, etc.**:
- The **I** format is used since an Immediate data element is needed. We use opcode 17 for the new instruction.
- We make **two** memory access for the new instruction: One to fetch the instruction (state 0) and one to write a data element to the memory (state 17).

**c)** There are three modified control signals (**ALUSrcA**, **IorD**, **MemWrite**), one new control signal (**Mwbussel**), one new next state signal (**NS4**) and one modified next state signal (**NS0**) in the control unit:

- **ALUSrcA** is 1 when it is state 2 or 6 or 8 or 16
  
  **ALUSrcA = S2 + S6 + S8 + S16**

- **S6**
  - **S2**
  - **S8**
  - **S16**

- **IorD** is 1 when it is state 3 or 5 or 17
  
  **IorD = S3 + S5 + S17**

- **MemWrite** is 1 when it is state 5 or 17
  
  **MemWrite = S5 + S17**

- **Mwbussel** is 1 when it is state 17
  
  **Mwbussel = S17**

- **NS4** is 1 when it is state 1 and Opcode 17 or state 16
  
  **NS4 = S1OPCDCD23 + S16**

- **S1**
  - **OPCDC23**
  - **S16**

- **NS4**

- **S5**
  - **S17**

- **Mwbussel**

- **S17**
  - **Mwbussel**
Q10) Assume that the EMY CPU high-level state diagram is modified to be able to run a new machine language instruction as follows:

![High-level state diagram]

- **a)** What is the new instruction? That is, determine its syntax, semantics, etc. If there is a new addressing mode that is not discussed in class, indicate so.

- **b)** i) Modify the EMY CPU datapath accordingly.

ii) How long does it take to run the new instruction? Explain.

- **c)** Assume that the Rs register above is R8 and its value is 10000000. Assume also that this new instruction is in location 400C00. Finally, assume that the DOImm is \((-8)_{10}\) for this instruction. Continue with the following table until the effect of the new instruction is visible on the architecture:

<table>
<thead>
<tr>
<th>Clock period</th>
<th>State</th>
<th>PC</th>
<th>IR</th>
<th>A</th>
<th>B</th>
<th>ALUout</th>
<th>R8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>-----</td>
<td>400C00</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>10000000</td>
</tr>
<tr>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>Continue</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td></td>
</tr>
</tbody>
</table>

- **d)** Modify the EMY CPU low-level state diagram accordingly and as done in class.

- **e)** Assume that the EMY CPU control unit is hardwired. Assume also that the opcode of the new instruction is 17. Modify the hardwired EMY CPU control unit accordingly, as done in class and as follows: Show two (2) signals that are modified or new. Again, just show the circuits for any two modified/new signals.

A10) **a)** On the high level state diagram, we see that we subtract 4 from register Rs then store to Rs. Afterwards, we check the previous value of Rs to see if it is 10000000 (the current value of Rs is FFFFFFC). If it is not, we move an address to PC by using the 2-byte signed PC-relative addressing mode. This is the BLP instruction “Branch Loop” instruction that has the following syntax and semantics:

**Syntax**: BLP Rs, Offset

**Semantics**:  
1) Rs ← Rs - 4
2) If Rs ≠ FFFFFFC then PC ← PC + (Offset + * 4)

**Format, etc.**: It uses the I format since an offset is used. Rt is not used.
It has seven arguments. Rs is a destination and source register argument using register addressing mode. Number “4” is an implied data element using the implied addressing mode. Rs is a source argument to be compared using the register addressing mode. FFFFFFC is an implied data element using the implied addressing mode. PC is implied to be the destination using implied addressing mode. The last operand is a memory address calculated by using the 2-byte PC-relative addressing mode.

We make one memory access for the new instruction: One to fetch the instruction (state 0)

b) i) The new instruction requires a number of changes in the datapath as shown below. The datapath can be modified in other ways though! SE means “Sign Extension” which sign extends the rightmost 16 bits of IR. “SE*4” means “Sign Extension times 4” which is the circuit that multiplies the sign extended rightmost 16 bits of IR by 4.

ii) CPI_{BLP} is 5 since we trace states 0, 1, 16, 17 and 18

c) The table is completed as follows:

<table>
<thead>
<tr>
<th>Clock period</th>
<th>State</th>
<th>PC</th>
<th>IR</th>
<th>A</th>
<th>B</th>
<th>ALUout</th>
<th>R8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>----</td>
<td>400C00</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>10000000</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>NS</td>
<td>NS</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>NS</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>400C04</td>
<td>BLP, R8, (-8)_{10}</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>NS</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>NS</td>
<td>NS</td>
<td>10000000</td>
<td>0</td>
<td>400BE4</td>
<td>NS</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>NS</td>
<td>NS</td>
<td>10000000</td>
<td>0</td>
<td>FFFFFFC</td>
<td>NS</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>NS</td>
<td>NS</td>
<td>10000000</td>
<td>0</td>
<td>400BE4</td>
<td>FFFFFFC</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>400BE4</td>
<td>NS</td>
<td>FFFFFFC</td>
<td>0</td>
<td>400BE4</td>
<td>NS</td>
</tr>
</tbody>
</table>

d) The modified EMY low-level state diagram is as follows:

- States 2 - 11: Same except
  - Step 2: ALUSrcB = 010
  - Step 4: RegDst = 00
  - Step 6: ALUSrcB = 000
  - Step 7: RegDst = 01
  - Step 8: ALUSrcB = 000

- Step 2: ALUSrcB = 010
- Step 4: RegDst = 00
- Step 6: ALUSrcB = 000
- Step 7: RegDst = 01
- Step 8: ALUSrcB = 000
e) Two **new** signals in the control unit, ALUSrcB2 and PCWriteCond2 are shown below:

- ALUSrcB2 is 1 when it is state 18
- ALUSrcB2 = S18
- S18 → ALUSrcB2

- PCWriteCond2 is 1 when it is state 18
- PCWriteCond2 = S18
- S18 → PCWriteCond2

Other new signals: RegDst1 and NS4
Modified signals: ALUSrcA, ALUSrcB1, ALUSrcB0, ALUop1, RegWrite, PCSrc0, NS2 and NS1

---

**Q11)** Assume that the EMY CPU **low-level** state diagram and the ** datapath** are modified to be able to run a **new** instruction as shown below.

**a)** Modify the EMY CPU **high-level** state diagram accordingly and as done in class. If a state is the same as the original, just write “Same.” What is the CPI of the new instruction? **Explain.**

**b)** What is the **new** instruction? Indicate only the following: The syntax, semantics, format and the memory accesses made.

**c)** Assume that the EMY CPU control unit is **hardwired**.

Modify the **hardwired** EMY CPU control unit accordingly, as done in class and as follows: Show **three** (3) signals that are modified or new.
A11) a) The modified high-level state diagram is as follows:

CPI₂ is 6 since we trace states 0, 1, 2, 3, 16 and 17.

b) On the high level state diagram, we see that we subtract register MDR from register B. This means we subtract a memory location content from register Rt. This is a subtract between a register and a memory location: SUBM:

Syntax: SUBM Rt, Disp(Rs)

Semantics: \( Rt \leftarrow Rt - M[Rs + Disp'] \)

Format, etc.:
- It uses the I format since a displacement is needed.
- We make two memory access for the new instruction: One to fetch the instruction (state 0) and one to read a data element from the memory (state 3).

c) Two new signals in the control unit, ALUSrcB2 and ALUSrcA1 and one modified control signal, RegWrite, are shown below:

ALUSrcB2 is 1 when it is state 16
\( ALUSrcB2 = S16 \)

ALUSrcA1 is 1 when it is state 16
\( ALUSrcA1 = S16 \)

RegWrite is 1 when it is state 4 or 7 or 17
\( \text{RegWrite} = S4 + S7 + S17 \)

ALUSrcB2
\( S16 \rightarrow ALUSrcB2 \)

ALUSrcA1
\( S16 \rightarrow ALUSrcA1 \)

Other new signals: SR4 and NS4. Modified signals: ALUop0 and NS0. Finally, ALUSrcA is renamed ALUSrcA0,