

HOMEWORK IV

DUE : November 3, 2009

READ :

- i) Related portions of Chapter 4 (except Sections 4.4 through 4.10)
- ii) Related portions of Appendix B
- iii) Related portions of Appendix C
- 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, datapath and low-level state diagram are modified for the JR instruction.

Modify the Control Unit of the EMY CPU for JR based on your answer in Homework 3. 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 **low-level state diagram** obtained in Homework 3 and then modify 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.

Modify the EMY CPU Control Unit for JAL based on your answer in Homework 3. 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 **low-level state diagram** obtained in Homework 3 and then modify 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 modification 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 Language 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 syntax and architectural operation are as follows :

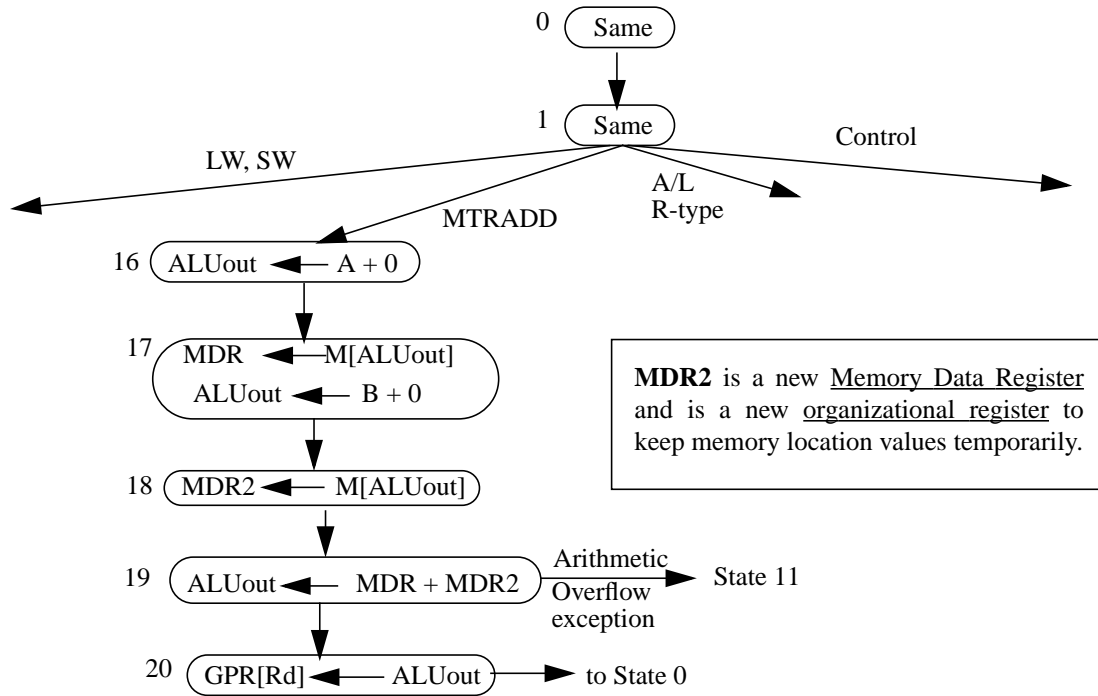
$$\text{MTRADD } Rd, (Rs), (Rt) \longrightarrow Rd \longleftarrow M[Rs] + M[Rt]$$

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_i of the instruction. Then, what is the CPI_i of the instruction?

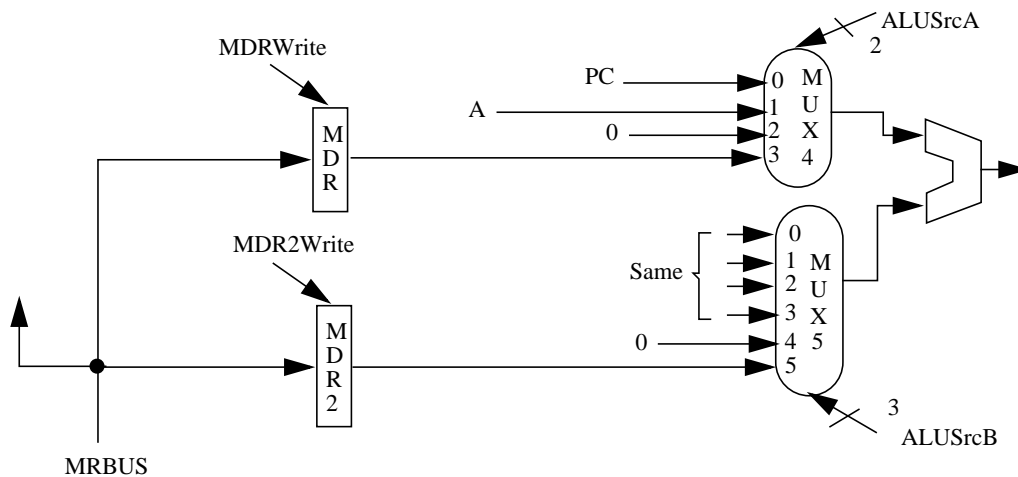
ii) Show the modified portion of the data unit together with control signals. Note that this is a CISC instruction since an A/L instruction accesses memory for data.

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_i 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:



iii) Assume that we run the following instruction :

400000 ABS R14, R15

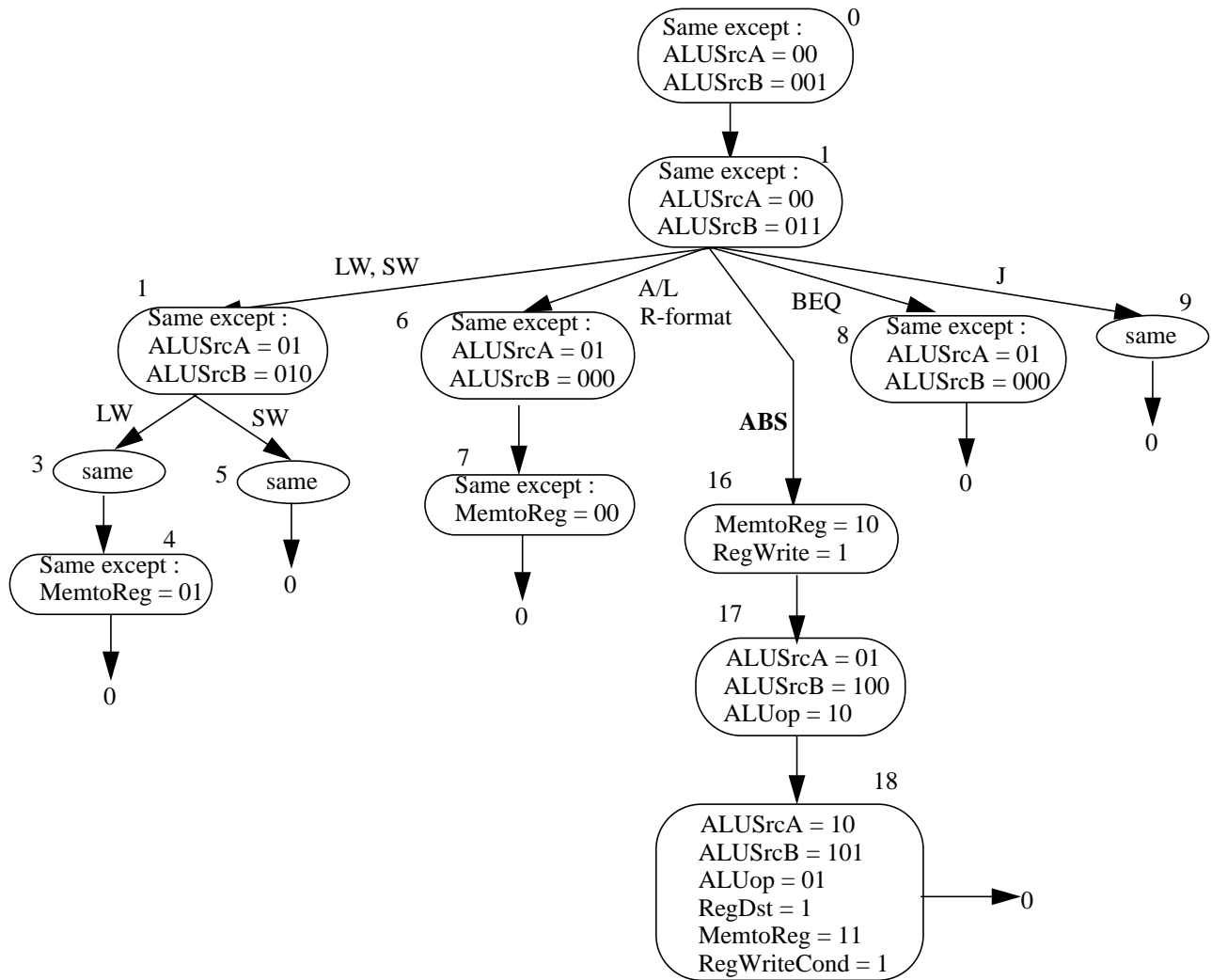
Show the values of registers and control signals for **five** clock periods for which continue with the following table :

Cp	State	PC	IR	ALUSrcA	RegWriteCond	ALUop	A	B	ALUout	R14	R15
Initial	---	400000	?	---	---	---	?	?	?	?	$(-6)_{10}$
1	0	NS	NS	00	0	00	?	?	?	NS	NS
...

A2) i) The new code is as follows :

400000 ABS R8, R9 # R8 ← |R9|

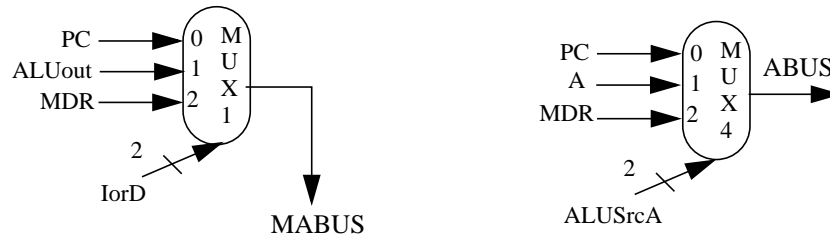
ii) The modified EMY low-level state diagram is as follows :



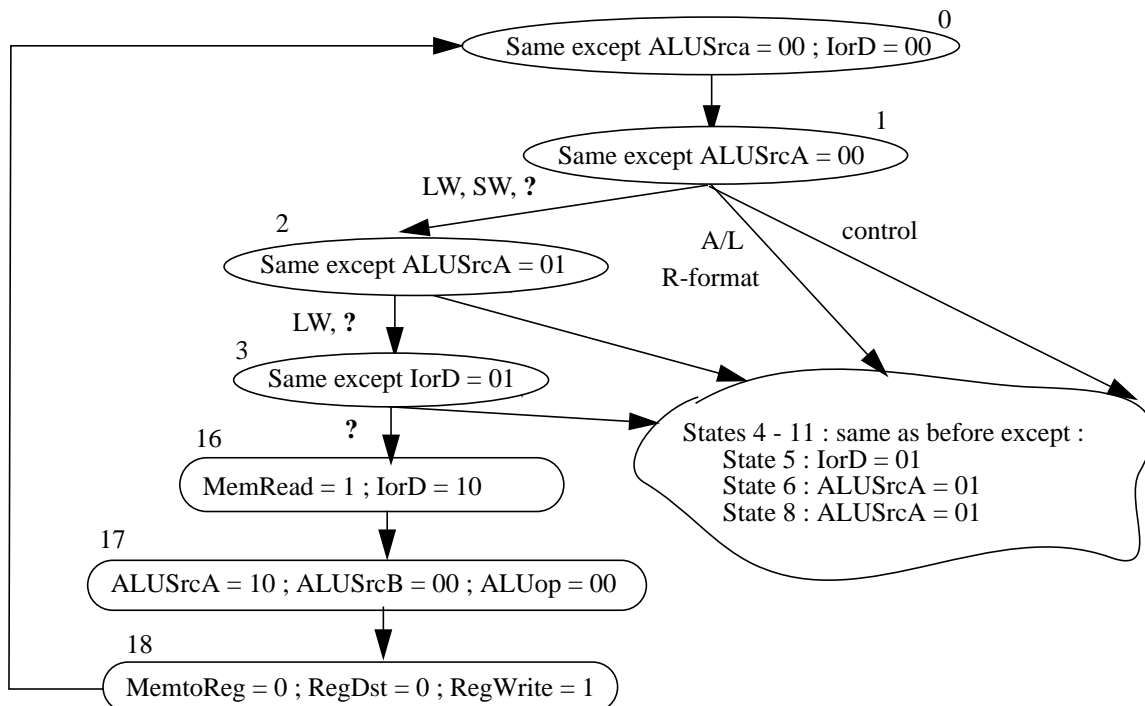
iii) The table with the values is as follows :

Cp	State	PC	IR	ALUSrcA	RegWriteCond	ALUop	A	B	ALUout	R14	R15
Initial	---	400000	?	---	---	---	?	?	?	?	$(-6)_{10}$
1	0	NS	NS	00	0	00	?	?	?	NS	NS
2	1	400004	ABS R14, R15	00	0	00	?	?	400004	NS	NS
3	16	NS	NS	00	0	0	$(-6)_{10}$	0	?	NS	NS
3	17	NS	NS	01	0	10	$(-6)_{10}$	0	400004	NS	NS
4	18	NS	NS	10	1	01	$(-6)_{10}$	0	1	NS	NS
5	0	NS	NS	00	0	00	$(-6)_{10}$	0	6	6	NS

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.



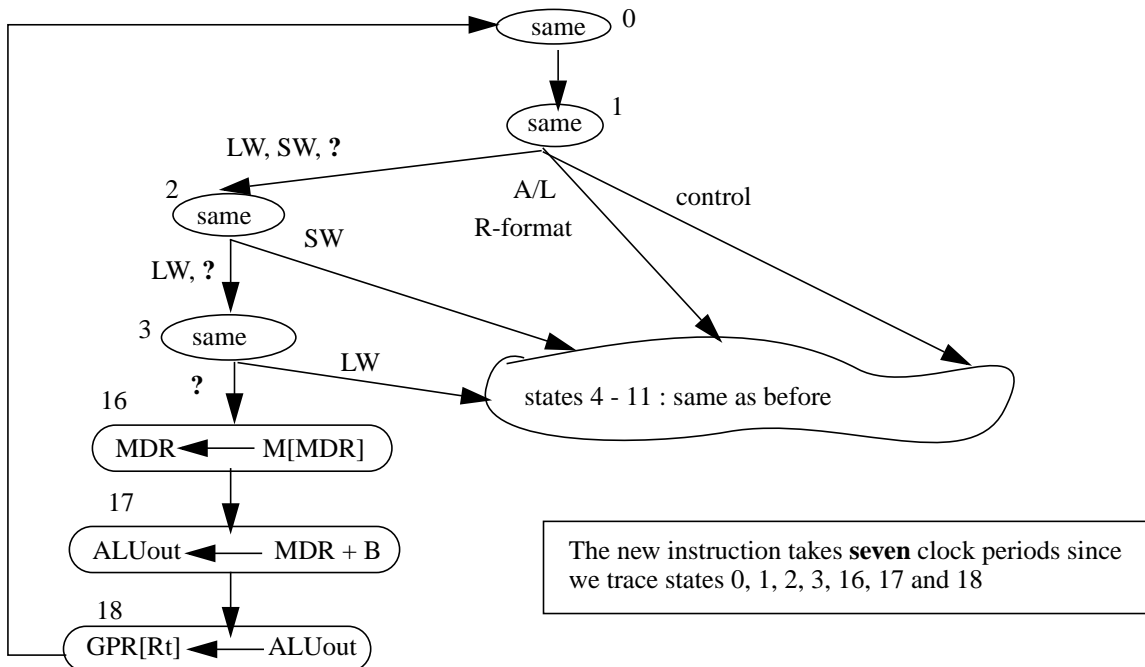
Obtain the corresponding **high-level** state diagram. How many clock periods does it take to run the new instruction ?

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 :



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

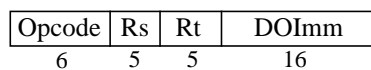
- | | | |
|---|---|-------------------------------------|
| $M[MDR] \leftarrow B$ | $ALUout \leftarrow MDR \text{ op } B$ | $PC \leftarrow MDR + 4$ |
| $ALUout \leftarrow MDR + B$ | $ALUout \leftarrow MDR \text{ op } 4$ | $PC \leftarrow MDR + DOImm^+$ |
| $ALUout \leftarrow MDR + 4$ | $ALUout \leftarrow MDR \text{ op } DOImm^+$ | $PC \leftarrow MDR + (DOImm^+ * 4)$ |
| $ALUout \leftarrow MDR + DOImm^+$ | $ALUout \leftarrow MDR \text{ op } (DOImm^+ * 4)$ | $PC \leftarrow MDR \text{ op } B$ |
| $ALUout \leftarrow MDR + (DOImm^+ * 4)$ | $PC \leftarrow MDR + B$ | 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 : $Rt \leftarrow M[M[Rs + Disp]] + Rt$

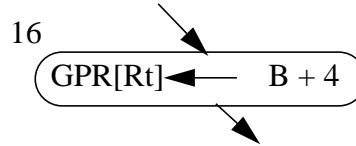
The format is the **I format** :



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 :

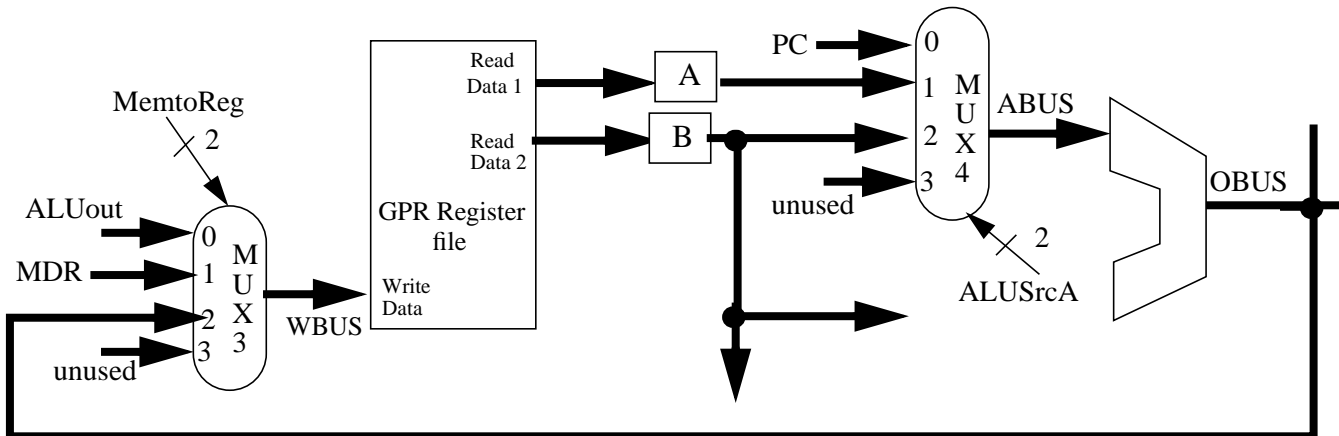


- a) Draw the modified portion of the EMY CPU datapath.
- b) Show the corresponding low-level state diagram.

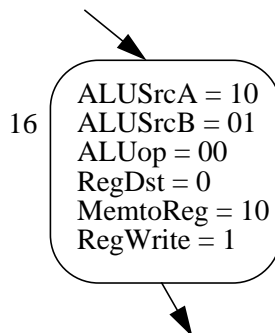
A4) a) We see that the ALU has to add B which always has Rt and 4. That is ABUS and BBUS are added such that they carry values B (Rt) and 4. In the datapath, already number 4 is connected to the BBUS MUX which is MUX5. Thus, we only need to connect register B to the ABUS MUX which is MUX4. Now, MUX4 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 $DOImm^+ \ll 2$ 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. The modified datapath is then as follows :



b) The corresponding low-level state :



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 :

clock period	State	PC	IR	R9	R12	M[4000F0]
Initial	---	4000F0	?	3F	?	YDI
1	0	NS	NS	NS	NS	NS
2	1	4000F4	YDI	NS	NS	NS
3	6	NS	NS	NS	NS	NS
4	7	NS	NS	NS	FFFFFFC0	NS

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

- a) What is the CPI_i 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 \longrightarrow 0000 0000 0000 0000 0000 0000 0011 1111
 FFFFFFFC0 \longrightarrow 1111 1111 1111 1111 1111 1111 1100 0000

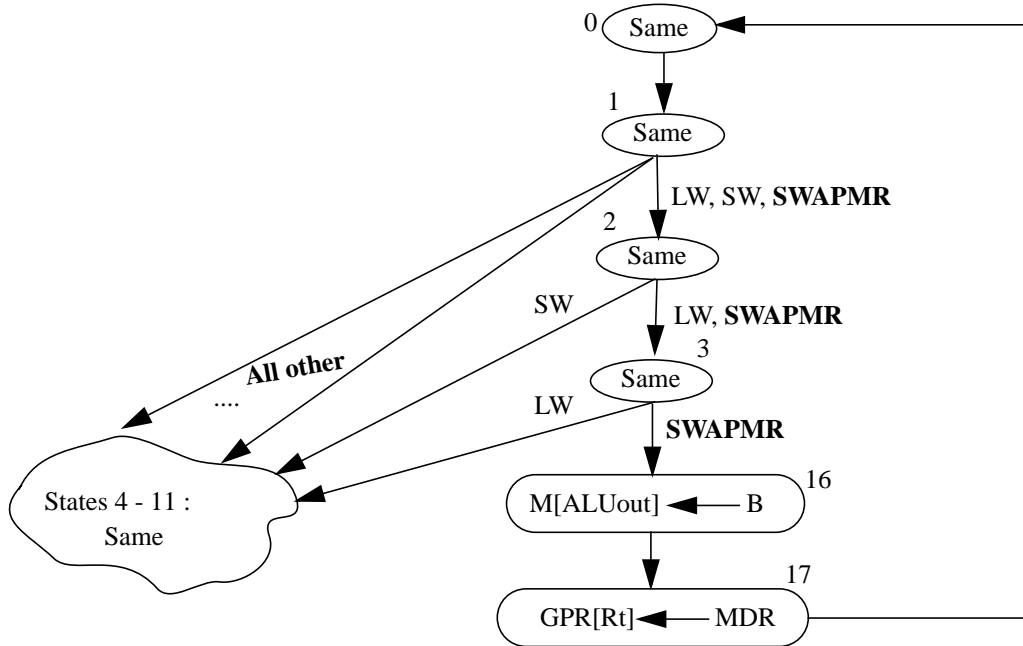
The two numbers are the complement of each other. Thus, we have a **complement** instruction with the R-format :

COMP Rd, Rs \longrightarrow Rd \longleftarrow \bar{R}_s

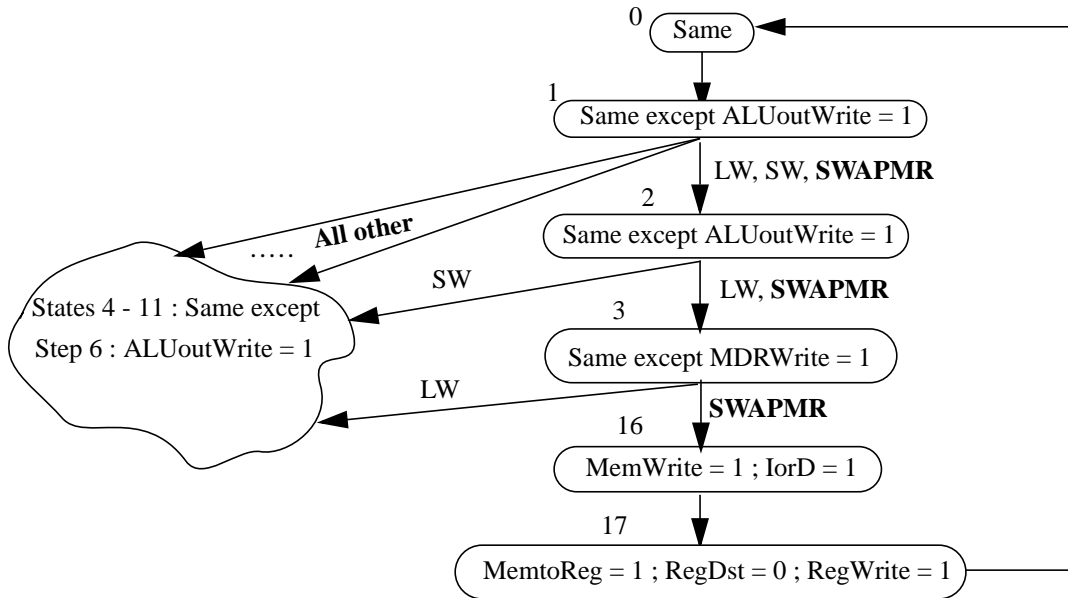
opcode	Rs	Unused	rd	shamt	function
6	5	5	5	5	6

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

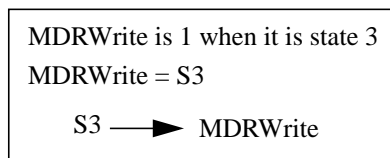
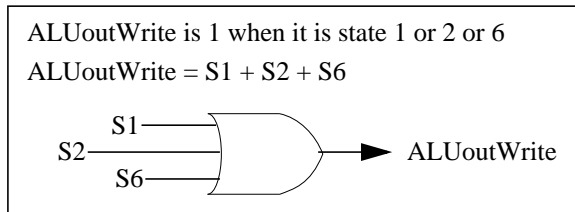
c) The only Data Unit change is in the ALU. The ALU must now complement the value on its ABUS input. In order to indicate that it is a complement, an ALUcontrol bit combination must be assigned the complement operation. We select combination 1000 for it :



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



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



A signal is modified :
NS0
 Another new signal :
NS4

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 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

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

iii) 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}** ?

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** and each memory access takes **one** clock period.

A7) 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. The execution time is 68ns which is computed as follows :

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

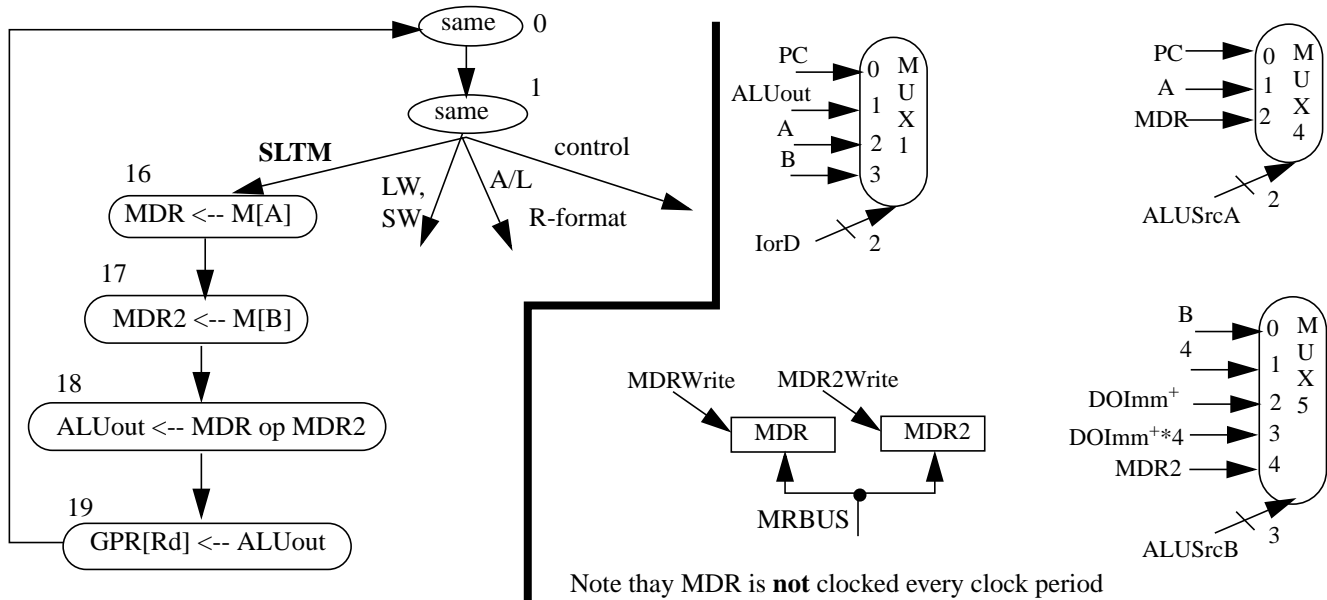
ii) The SLTM instruction replaces the first **three** instructions :

```

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

```

iii) 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

iv) 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 $57 * 1 = 57ns$. 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 !

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)10    # 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 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 :

LW : 0, 1, 2, 3, 4 => 5 cp BNE : 0, 1, 16 => 3 cp ADDI : 0, 1, 16, 17 => 4 cp
 ADD : 0, 1, 6, 7 => 4 cp SW : 0, 1, 2, 5 => 4 cp

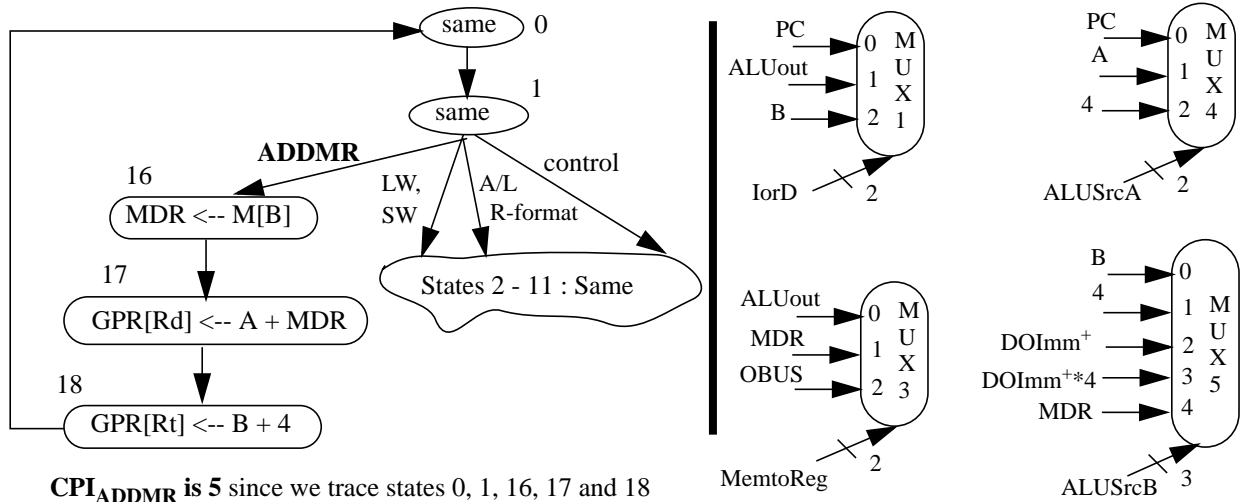
The number of clock periods to run the 12 instructions is $4 + 2(5 + 4 + 4 + 4 + 3) + 4 = 48$ clock periods.

clock period = $\frac{1}{1 \times 10^9} = 1 \times 10^{-9} \text{ sec} = 1 \text{ ns}$ Execetime = # of clock periods for the code * clock period
 Execetime = $48 * 1 = 48 \text{ ns}$

ii) The ADDMR instruction replaces **three** instructions :

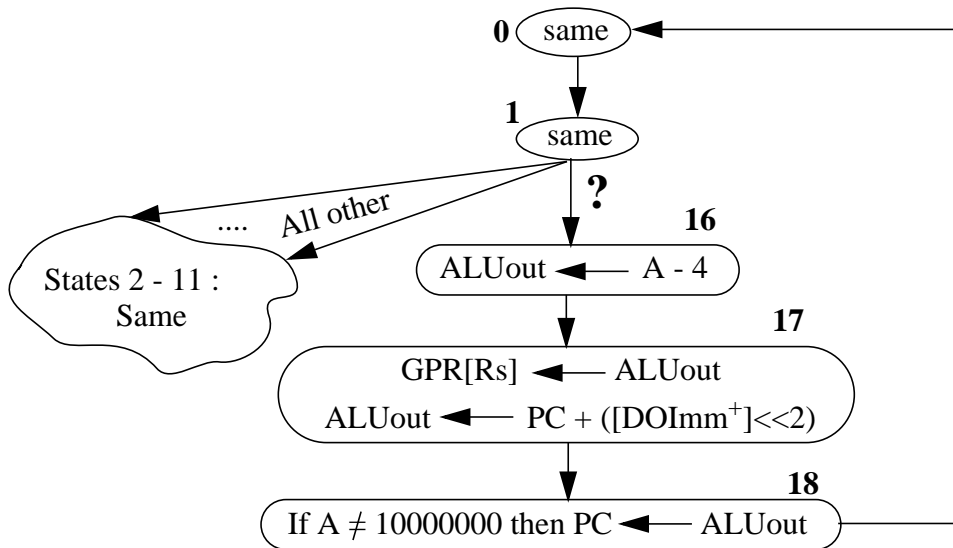
400000	ADDI	R8, R0, 0	# Initialize R8 to 0
400004	ADDMR	R8, R8, (R10)++	# Add an array A element pointed by R10 to R8 then update R10
400008	ADDI	R11, R11, (-1) ₁₀	# The loop-end counter, R11, has 2 initially
40000C	BNE	R11, R0, (-3) ₁₀	# If not the end, go back to location 400004
400010	SW	R8, 0(R10)	# 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 :



iv) We execute the following eight instructions : $ADDI + 2(ADDMR + ADDI + BNE) + SW$. 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 $32 * 1 = 32 \text{ ns}$.

Q9) Assume that the EMY CPU **high-level** state diagram is modified to be able to run a **new** machine language instruction as follows :



- 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 :

Clock period	State	PC	IR	A	B	ALUout	R8
Initial	----	400C00	?	?	?	?	10000000
.....		Continue

- 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.

A9) 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 FFFFFFFC). 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) $Rs \leftarrow Rs - 4$

2) If $Rs \neq FFFFFFFC$ then $PC \leftarrow PC + (Offset^+ * 4)$

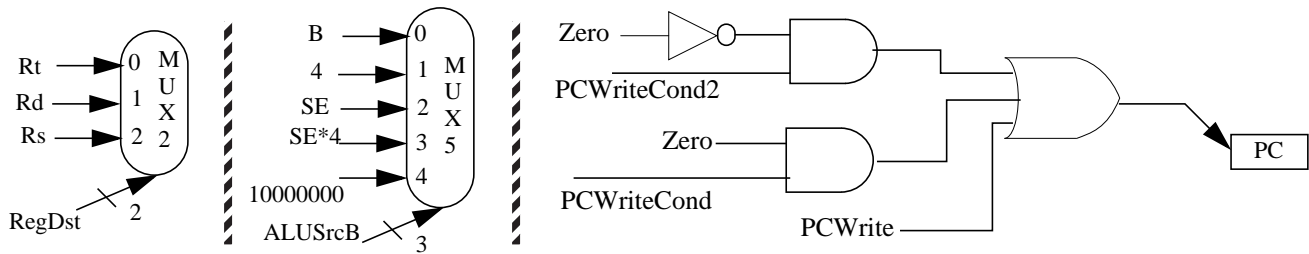
Format, etc. : It uses the **I** format since an offset is used. Rt is not used

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. FFFFFFFC 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 modi-

fed 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.

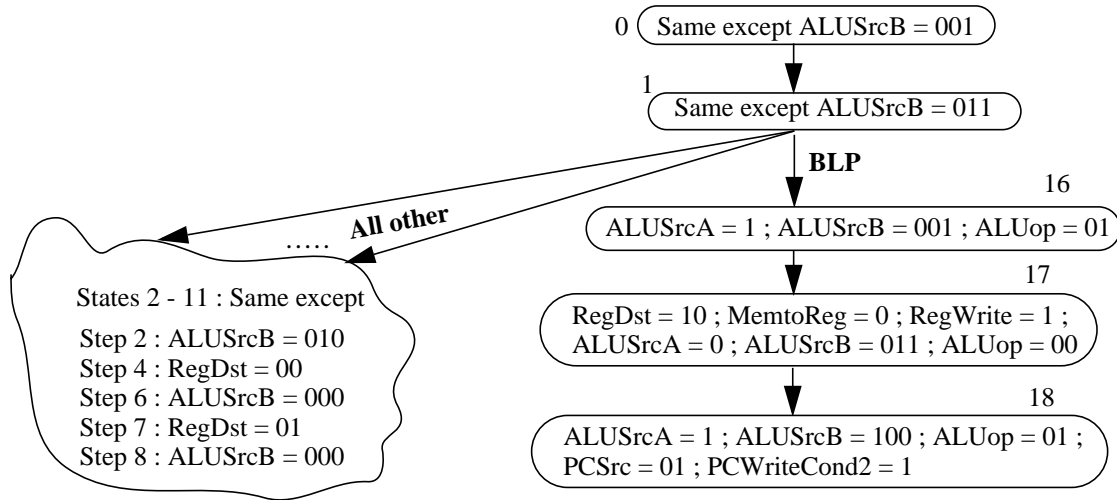


CPI_{BLP} is 5 since we trace states 0, 1, 16, 17 and 18

c) The table is completed as follows :

Clock period	State	PC	IR	A	B	ALUout	R8
Initial	----	400C00	?	?	?	?	10000000
1	0	NS	NS	?	?	?	NS
2	1	400C04	BLP, R8, (-8) ₁₀	?	?	?	NS
3	16	NS	NS	10000000	0	400BE4	NS
4	17	NS	NS	10000000	0	FFFFFFC	NS
5	18	NS	NS	10000000	0	400BE4	FFFFFFC
6	0	400BE4	NS	FFFFFFC	0	400BE4	NS

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



e) Two new signals in the control unit, ALUSrcB2 and PCWriteCond2 are shown below :

