Rubric for evaluating Complex Engineering Problem

Course: **(CS 221) Computer Organization & Design** (CLO 2 – C3) Department: **CIS**
Class: **SE-CIS** Batch: **2020** Academic Session: **Spring 2022**
Topic: **Processor Design**

Group Members:

Student No.	Name	Roll No.	Batch
S1	**Hiba Khan**	**CS-018**	**2020**
S2	**Syed Aali**	**CS-053**	**2020**

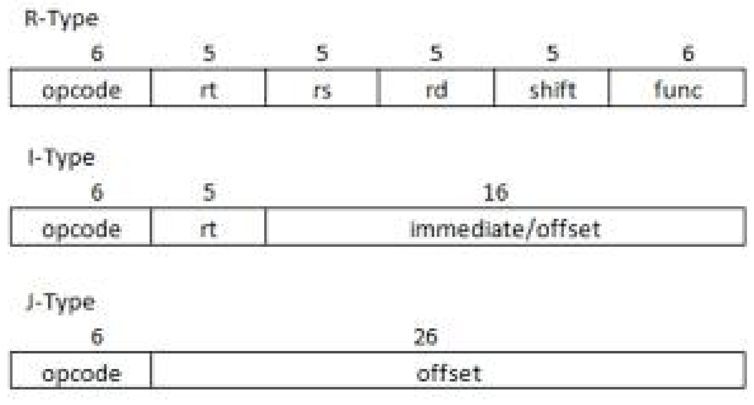
CRITERIA AND SCALES			**Marks**		
			S1	**S2**	**S3**
Criterion 1: Depth of fundamental knowledge required for the design? (CPA-1)					
0-1	**2-3**	**4-5**			
Proposed solution is inacceptable, showing student has ambiguous understanding of fundamentals.	*Proposed solution is acceptable to a certain extent showing student has good understanding of fundamentals.*	*Proposed solution is outstanding showing student has clear underarming of fundamentals.*			
Criterion 2: To what extent student's design is innovative? (CPA -2)					
0-1	**2-3**	**4-5**			
Proposed design is same as that of other student(s)	*Proposed design is different from others.*	*Proposed design is innovative.*			
Criterion 3: To what extend design can be extended? (CPA-3) (as evident from implementation of part **b** of assigned task)					
0-1	**2-3**	**4-5**			
Students were not able to modify the existing design at all or slightest changes were made.	*Students were able to modify the existing design, but it contained some mistakes.*	*Students successfully modified the existing design.*			
Criterion 4: To what extent design is working?					
0-1	**2-3**	**4-5**			
Proposed design is not working correctly or partially working.	*Proposed design is working but it contained some mistakes.*	*Proposed design is working correctly in all aspects.*			
Criterion 5: To what extent the student was able to express the cognitive process required in the design? (Report Writing Skills)					
0-1	**2-3**	**4-5**			
Only diagrams are presented. The documentation is poor.	*Informal lexis is used. The documentation is average.*	*Report is accompanied with formal technical write-up. Work presentation is excellent.*			
Criterion 6: How student responded to viva voce?					
0-1	**2-3**	**4-5**			
Student had no idea of design. He was unable to answer most of the questions.	*Student was involved in the design process and answered many questions.*	*Student actively participated in the design process and answered most of the questions.*			

Teacher's Signature: ________________

What is MIPS SCDP?

MIPS is a family of RISC (reduced instruction set computer) instruction set architectures. It has a single cycle data path (SCDP) which is divided into two components, the main data path and the control unit (CU). The SCDP contains structures such as memory, registers, ALUs, instruction fetching unit and multiplexers.

The R-Type, I-Type and J-Type are the three types of instructions that MIP's follows.



PART A

DATAPATH ELEMENTS

The following components are present in the Data path:

Memory:

MIPS has two separate memories for data and instruction. It is a byte addressable architecture where each byte has a unique address. However when we map our MIPS SCDP In logisim, we have to remember the memory here is actually word addressable instead of byte addressable.

Instruction Memory:

Instruction Memory is responsible for holding the instructions of program. It receives the address of instruction from PC (program counter). The address and instruction are of 32-bit in MIPS.

Data memory:

Data Memory holds all the data that comes with a program. It has ports to read and write control signals that tell it when to read and when to write. The address is of 24-bit where as the read/write data are of 32-bits.

Register File:

The register file contains the 32 MIPS general purpose registers. There are two register read data ports and one register write data port which are both 32 bits wide. The RegWrite control signals tells us whether the data received in the write data port will be written or not. If the write enable is 1, the register file writes thedata into the specified register.

ALU:

The Arithmetic Logic Unit performs various arithmetic and logic operations. The operations are performed one at a time and selected using a 2x1 MUX. The select line is named as ‘ALU Operation’. It has two inputs as data source, and two outputs, one for result and the other is named as zeroflag. Zero flag is ON whenever the ALU result is zero or both inputs are identical.

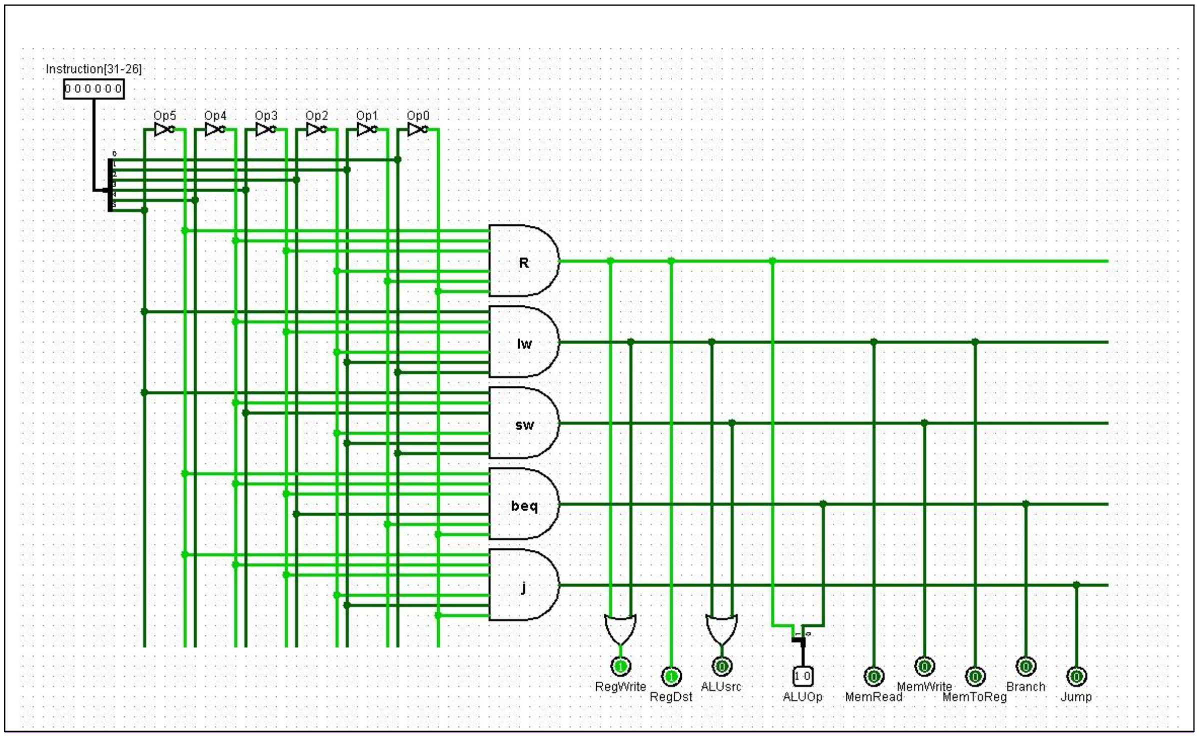
Control Unit:

The control unit is implemented using the programmable logic arrays. It is given the current instruction from instruction memory and tells us how to execute the instruction provided. In particular, it produces multiplexer selection, register enable, and memory read and write signals to control the operation of the data path. The names of control signals are RegWrite, RegDst, ALUsrc, ALUOp, MemRead, MemWrite, MemToReg, Branch and Jump. Only the ALUOp instruction is of 2-bits and it is sent to the ALU control unit.

These signals are decided based on the datapath, ALUOp is user defined which is assumed as:

The Instruction	ALUOp
R- Type	10
LW/SW	00
BEQ	01

	Instruction[5-0]	r	lw	sw	beq	j
Input	Op_5	0	1	1	0	0
	Op_4	0	0	0	0	0
	Op_3	0	0	1	0	0
	Op_2	0	0	0	1	0
	Op_1	0	1	1	0	1
	Op_0	0	1	1	0	0
Output	RegWrite	1	1	0	0	0
	RegDst	1	0	X	X	X
	ALUsrc	0	1	1	0	X
	$ALUOp_1$	1	0	0	0	X
	$ALUOp_0$	0	0	0	1	X
	MemRead	0	1	0	0	0
	MemWrite	0	0	1	0	0
	MemToReg	0	1	X	X	X
	Branch	0	0	0	1	0
	Jump	0	0	0	0	1



ALU Control:

The ALU Control distinguishes between various R-Type instructions. It has two inputs, one is the control signal ALUOp, and the other is 6-LSB bits of instruction. The output is of 3-bits named as ALU operation, which is further used as the input to ALU.
A truth table for predefined ALU Instructions has been shown below.

Instruction	function						ALUOp		ALU Operation		
	F_5	F_4	F_3	F_2	F_1	F_0	$ALUOp_1$	$ALUOp_0$	A_2	A_1	A_0
and	1	0	0	1	0	0	1	0	0	0	0
or	1	0	0	1	0	1	1	0	0	0	1
add	1	0	0	0	0	0	1	0	0	1	0
sub	1	0	0	0	1	0	1	0	1	1	0
slt	1	0	1	0	1	0	1	0	1	1	1
lw/sw	X	X	X	X	X	X	0	0	0	1	0
beq	X	X	X	X	X	X	0	1	1	1	0

Instruction Fetching Unit:

This unit is used to fetch address of next instruction in the PC. The next instruction address depends on the program which could be either sequential, branch and jump.

For sequential execution we have the formula:

$$PC = PC + 4$$

Since in Logisim we have word addressable format instead of bytes it is calculated as:

$$PC = PC + 1$$

BTA (branch target address) is calculated by the formula:

$$BTA = (PC+1) + offset$$

JTA (jump target address) is calculated as:

$$JTA = (PC+1)\,[31:28]\ ||\ PDA$$

Where PDA is pseudo direct address calculated as:

$$PDA = JTA$$

The selection between three options is done using two 2x1 multiplexers in our circuit.

INTERJOINING THE DATAPATH ELEMENTS

All the data path elements are to be joined together. For viewing outputs, tunnels have been used.

All done on the circuit files*

PROGRAM

We had to first choose a program that integrated all the parts of SCDP instructions and thus we chose the program

```
If q==0:
    p[0] = q-w
else:
    p[0] = q+w
```

Assuming our variables are stored in registers:
q = $5
w = $6
p = base address in $9

The base address is 30h and the program is stored at address 5h

Assembly Code:

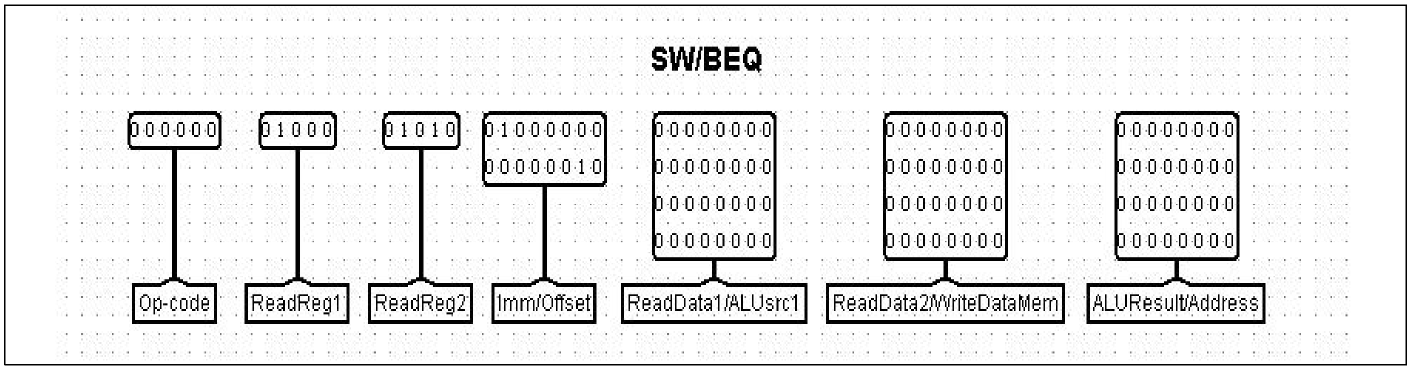
Address	Instruction	Comment
5h	beq $5, $0, Else	Checking for q==0 if yes go to else
6h	add $7, $5, $6	Adding q + w and storing in p
7h	J End	Jumping to store word instruction
8h	Else: sub $7, $5, $6	Subtracting q-w and storing in p
9h	End: sw $7, 0($9)	Storing in p[0]

Machine Code:

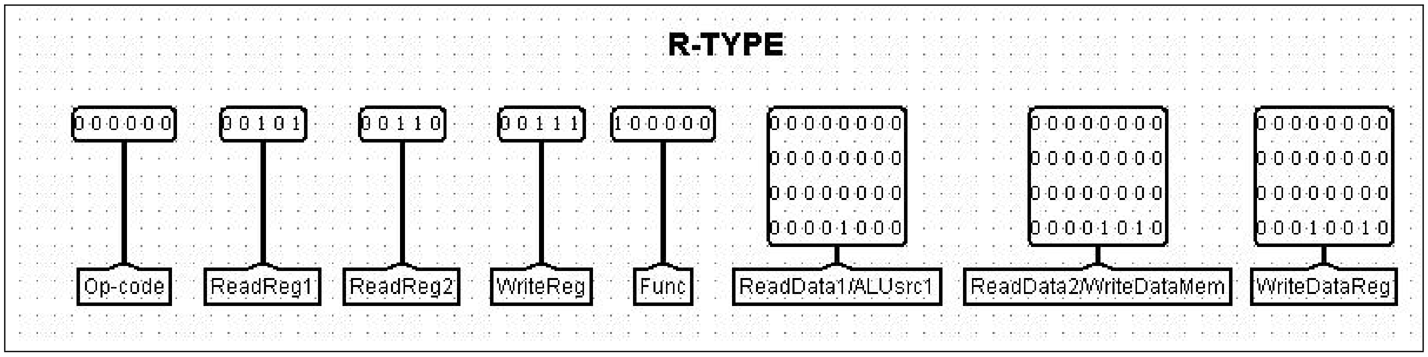
Address	Instruction Format	Hex Code
5h	4 | 5 | 0 | 2	10A4002
6h	0 | 7 | 5 | 6 |0 | 32	00A63820
7h	2 | 9	08000009
8h	0 | 7 | 5 | 6 | 0 | 34	00A63824
9h	43 | 9 | 7 | 0	AD270000

SIMULATION

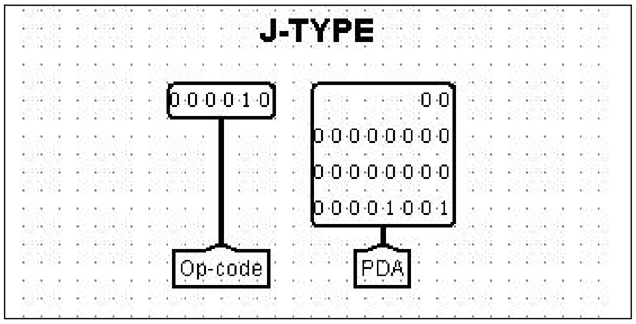
beq $5, $0, Else



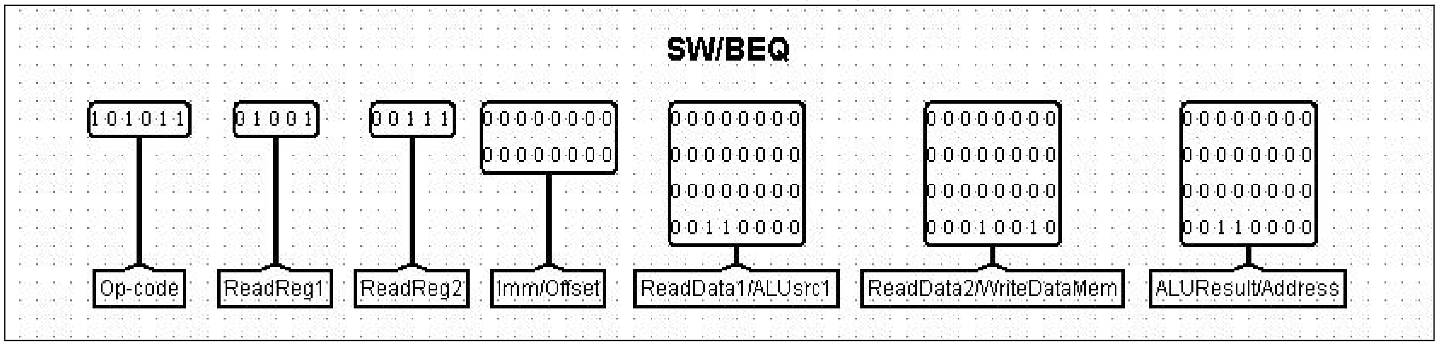
add $7, $5, $6



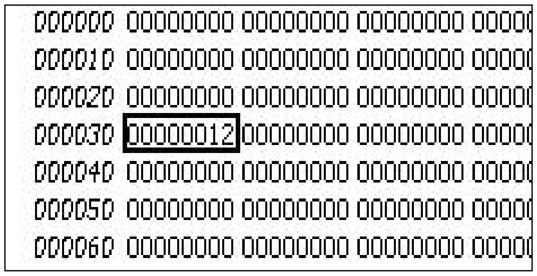
J End



End: sw $7, 0($9)



Answer of program is stored in memory variable p



```
000000 00000000 00000000 00000000 0000
000010 00000000 00000000 00000000 0000
000020 00000000 00000000 00000000 0000
000030 00000012 00000000 00000000 0000
000040 00000000 00000000 00000000 0000
000050 00000000 00000000 00000000 0000
000060 00000000 00000000 00000000 0000
```

PART B

Modifications In SCDP for LWM

We know:

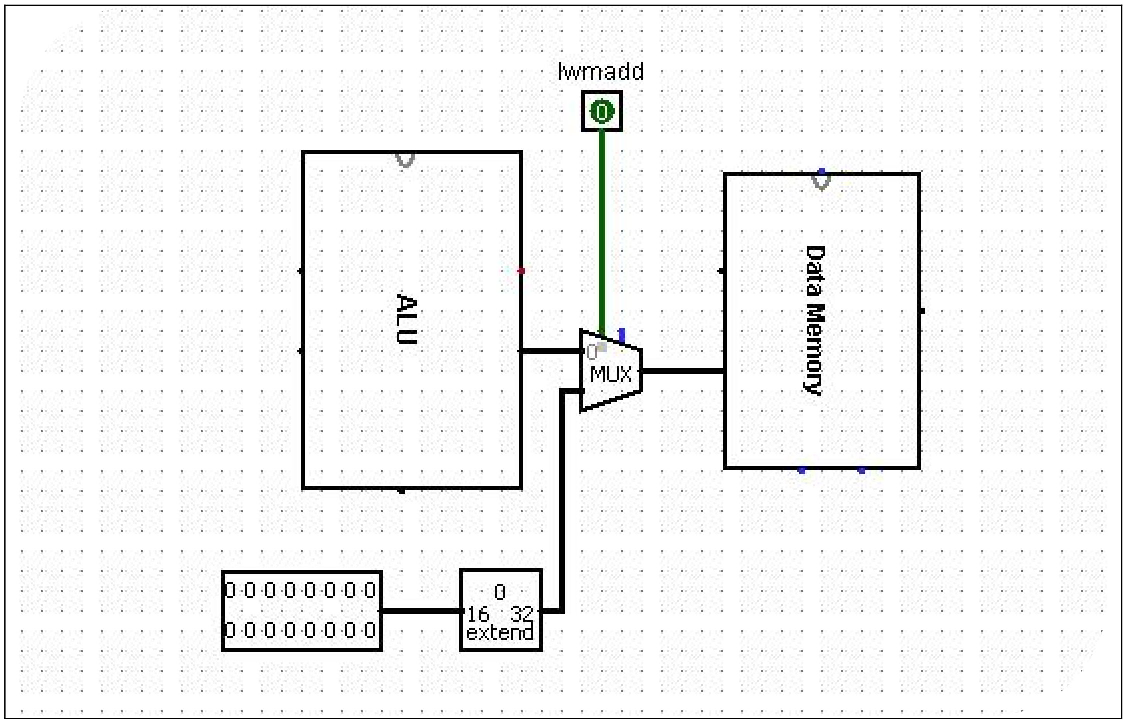
Lwm rt, address

The instruction LWM follows i-format type of instructions and that the address is of 16-bit which is stored in immediate/offset field of the format.
This is extended to 32-bits to make an address in data memory.
Since the $rs field isn't used its value is taken as 0.
The instruction would look something like:

Op-code	0	$rt	Immediate/offset

We are assuming our OP-Code to be 12 (001100).

In order to implement both lwm and lw/sw, there are two choices for data memory address, selecting one of them is done by adding a 2x1 MUX on the address line. The name of select line is lwmadd.

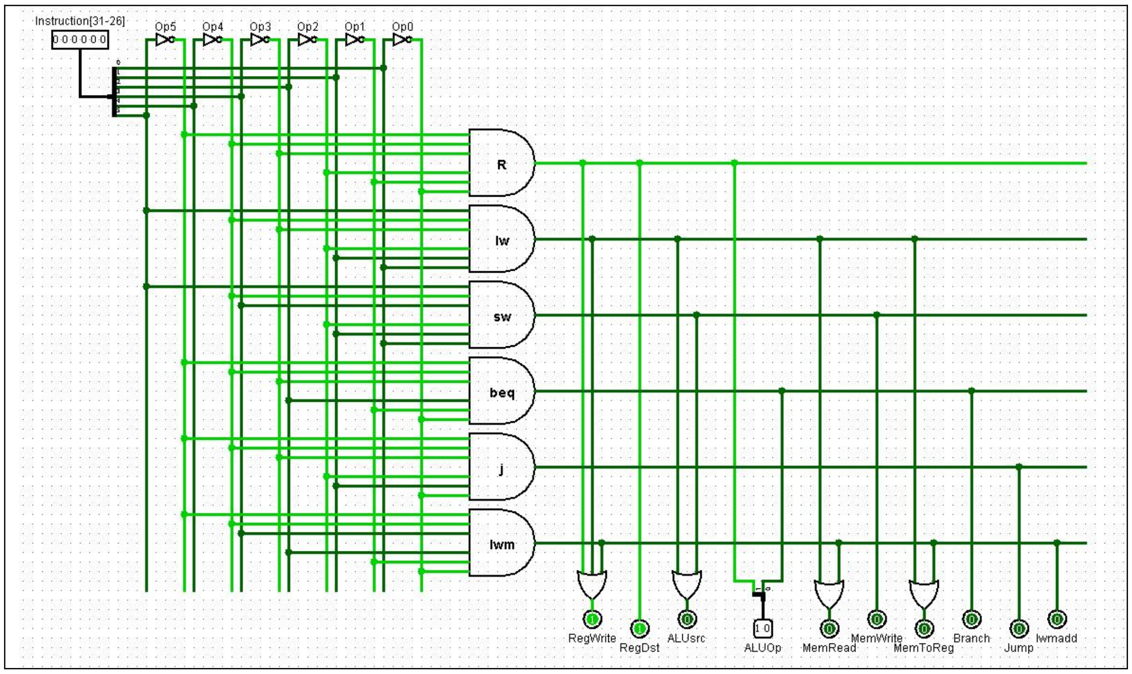


The values of control signals required for lwm are as follows and several modifications have been made in the control unit.

Control Signals	Values
RegDst	0
RegWrite	1
ALUSrc	X
ALUOP	XX
MemRead	1
MemWrite	0
MemToReg	1
Branch	0
Jump	0
NewAddr	1

Control unit:

	Instruction[5-0]	R-Type	lw	sw	beq	j	lwm
Input	Op_5	0	1	1	0	0	0
	Op_4	0	0	0	0	0	0
	Op_3	0	0	1	0	0	1
	Op_2	0	0	0	1	0	1
	Op_1	0	1	1	0	1	0
	Op_0	0	1	1	0	0	0
Output	RegWrite	1	1	0	0	0	1
	RegDst	1	0	X	X	X	0
	ALUsrc	0	1	1	0	X	X
	$ALUOp_1$	1	0	0	0	X	X
	$ALUOp_0$	0	0	0	1	X	X
	MemRead	0	1	0	0	0	1
	MemWrite	0	0	1	0	0	0
	MemToReg	0	1	X	X	X	1
	Branch	0	0	0	1	0	0
	Jump	0	0	0	0	1	0
	ModAddr	X	0	0	X	X	1



(other datapath elements will remain same, no modification is required)

PROGRAM

The program is same as part a, just the new instruction is run on the modified circuit. To check lwm, the result stored in part a is retrieved back to $23.

Assembly Code:

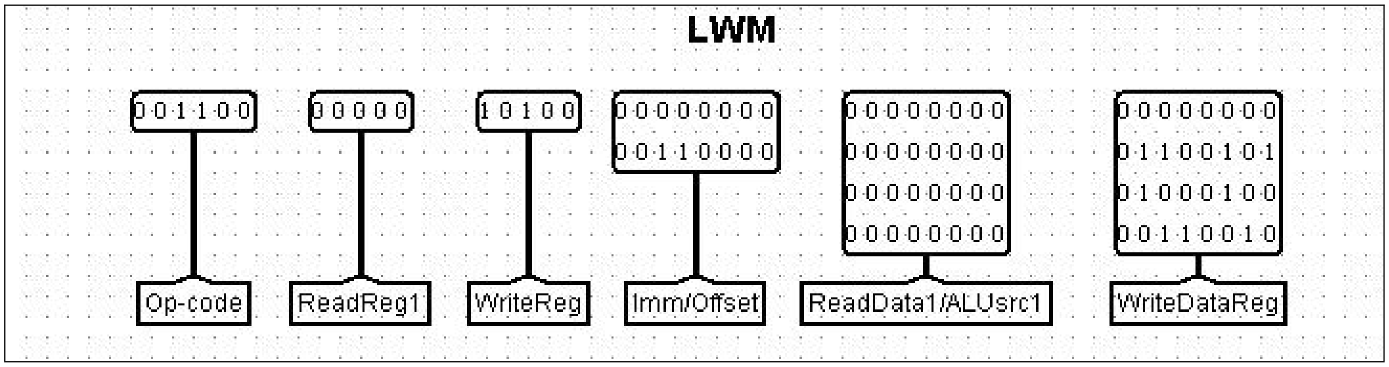
Lwm $20, 32

Machine Code:

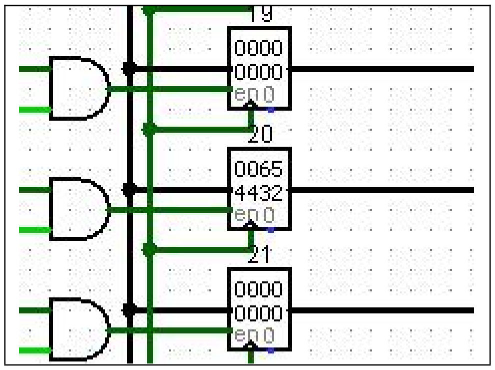
12	0	20	48

In hex: 30140030

SIMULATION



Data loaded successfully



19
0000
0000
en 0
20
0065
4432
en 0
21
0000
0000
en 0