NORTH SOUTH UNIVERSITY

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

Project

Computer Organization and Architecture

CSE332

Summer 2021

Project Part 1: ISA submission. 12/20

Course : CSE332.

Group #: 3.

Date: 23-July-21

Submitted to: Dr. Tanzilur Rahman.

Group Members:

Name	ID
Istiak Ahmed Sheam	1911139642
Md Ulfat Tahsin	1913057642
Md Saeem Hossain Shanto	1912218642
Salman Meem Sahel	1813077042

In this project, we are going to design a 14-bit processor as per the given instruction.

Our processor characteristics are:

- RISC Type processor (single-cycle CPU).
- Load-Store architecture.
- There are 2 operands (each 4-bit long).
- Operands are register-based.
- There are three formats of data our processor will receive:
  - ➔ R-type.
  - ➔ I-type.
  - ➔ J-type.
- We have 16 registers in the register file (each of 4-bit length).
- We are going to handle 16 different types of operations. And those include:
  - ➔ Arithmetic.
  - ➔ Logical.
  - ➔ Data transfer.
  - ➔ Conditional branch.
  - ➔ Unconditional jump.
  - ➔ External communication.

Register File:

We have 16 registers in the register file as given below:

Serial # of the register	**Name of the register**	**Used for**	**Value assigned (4-bit each)**
0	$zero	Constant value 0	0000
1	$s0	Save	0001
2	$s1	Save	0010
3	$s2	Save	0011
4	$s3	Save	0100
5	$t0	Temporary	0101
6	$t1	Temporary	0110
7	$t2	Temporary	0111
8	$t3	Temporary	1000
9	$t4	Temporary	1001
10	$t5	Temporary	1010
11	$t6	Temporary	1011
12	$t7	Temporary	1100
13	$t8	Temporary	1101
14	$t9	Temporary	1110
15	$t10	Temporary	1111

Data formats & Operations:

R-type format:

Op-code	**rs1**	**rs2**	**shamt**
4-bit	4-bit	4-bit	2-bit

Operations:

Category	**Op-code**	**Operation**	**Example**	**Description**
Arithmetic	0000	add	add $s0, $t0	$s0−>$s0 + $t0. The register rs1 will consist of the value obtained after summing the values in $s0 and $t0. The $s value is thus overwritten.
Arithmetic	0001	sub	sub $s0, $t0	$s0−>$s0 - $t0 The register rs1 will consist of the value obtained after subtracting the values in $s0 from $t0. The $s0 value is thus overwritten.
Logical	0010	AND	AND $s0, $t0	$s0−>$s0 AND $t0. The register rs1 will consist of the value obtained after the logical AND operation of value in $s0 and $t0. Thus, the rs1 value is overwritten.
Logical	0011	OR	OR $s0, $t0	$s0−>$s0 OR $t0. The register rs1 will consist of the values obtained after the performing of logical OR operation between the values stored in the registered $s0 and $t0. The value in $s0 is thus overwritten.
Logical	0100	NOR	NOR $s0, $t0	$s0−>$s0 NOR $t0. The register rs1 will consist of the values obtained after the performing of logical NOR operation between the values stored in the registered $s0 and $t0. The value in $s0 is thus overwritten.

I-type format:

Op-code	**rs1**	**immediate**
4-bit	4-bit	6-bit

Operations:

Category	**Op-code**	**Operation**	**Example**	**Description**
Arithmetic	0101	addi	addi $s0, 5	$s0−>$s0 + constant. The register $s0 will consist of the value obtained after summing the values in $s0 and the constant. The $s0 value is thus overwritten.
Logical	0110	ANDi	ANDi $s0, 5	$s0−>$s0 AND constant. The register rs1 will consist of the value obtained after performing the logical AND operation with the values in $s0 and the constant. The $s0 value is thus overwritten.
Logical	0111	ORi	ORi $s0, 3	$s0−>$s0 OR constant. The register $s0 will consist of the value obtained after performing the logical OR operation with the values in $s0 and the constant. The $s0 value is thus overwritten.
Conditional branch	1000	beq	beq $s0, ~~4($t0)~~ Label	If ($s0 == $t0) jump to the labelled location 4($t0). Otherwise, jump to another label. The label is the position where the target instruction is located.
Data transfer	1001	lw	lw $s0, ~~4($s1)~~ Label	This instruction will go to the memory in $s1, get the value and bring it to the register $s0 as per the mentioned offset label.

incorrect

incorrect

4

beq S0, S1, L1

incorrect	Data transfer	1010	sw	sw $s0, ~~4($s1)~~ Label	This instruction will go to the memory in $s1, store the value that was in the register $s0 as per the mentioned offset label.
	Shift left logical	1011	sll	sll $s0, 4	$s0−>$s0 << constant. This will shift left the value of the content in $s0 to 4 position maximum (0 to 4) and overwrite the value in $s0 with the new one.
	Shift right logical	1100	srl	srl $s0, 3	$s0−>$s0 >> constant. This will shift right the value of the content in $s0 to 4 position maximum (0 to 4) and overwrite the value in $s0 with the new one.
incorrect	External communication	1101	IN	IN $t0, ~~5~~ Label	It is used to scan data into the device with the help of any input device. For example: a keyboard.
incorrect	External communication	1110	OUT	OUT $s0, ~~5~~ Label	It is used to visualize/print data from the device with the help of any output device. For example: a seven-segment display.

\

J-type format:

Op-code	**Target**
4-bit	10-bit

Operation:

Category	**Op-code**	**Operation**	**Example**	**Description**
Unconditional jump	1111	Jump	j 4(~~$t0+2~~)	Jump Memory [$t0 + constant]. Jump to the level

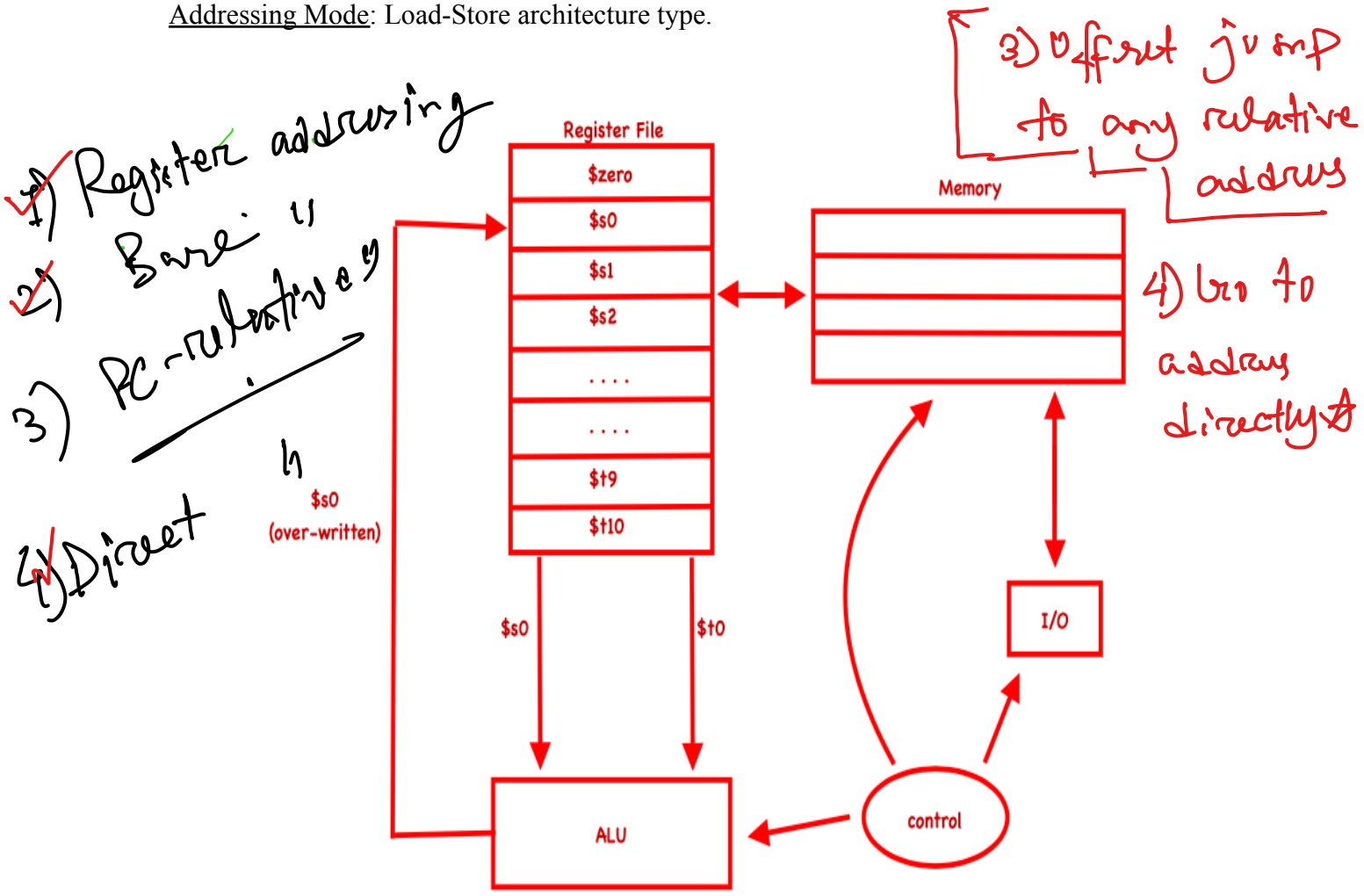
incorrect

target

Mail:

1) Appointment for project

Addressing Mode: Load-Store architecture type.



Notes on the different types of registers used in our project:

1) Temporary Register ($t): it is used to hold intermediate results. It is required to be initialized before it can be used. There can be one or more temporary registers as per need. These can store temporary results when the ALU performs any kind of temporary or logic operations. They can store operands or addresses that are part of an existing instruction.
2) Save Registers ($s): it is used to hold different values or to be specific location of the values needed to execute any instruction.
3) Zero Register ($zero): it is hard wired with the ground that always provides a zero value. Can be used for multiple tasks like copying, performing inverse operations etc.