ISA Design

(Project Milestone 01)

PREPARED FOR

CSE332 **Section 09**

Computer Organization and Architecture

PREPARED BY

Group 06

Arjun Karmaker (2031670642)

Jannatul Ferdous (2031530642)

Tanvir Ahmed Chowdhury (2031227642)

Tasnova Tabassum (2121463642)

ISA (Instruction Set Architecture)

Introduction:

As we are a team of computer architects employed in a vendor company tasked with a goal to design a new **10 bit single-cycle CPU** that has separate Data and Instruction Memory, we designed a general purpose ISA which should be able to run the provided general programs.

Objective:

Designing an ISA focusing on the following categories of benchmark programs:

- Simple arithmetic & logic operations (ADD / AND)
- Programs that require checking conditions (if-else conditions)
- Loop type of programs (for loops)

Input/Output Operations:

This computer system will be able to connect with the display unit (ex: seven-segment display) to display results or any data using the **OUT** instruction. Input will come from a keyboard or something similar to get input from the user using the **IN** instruction.

Design Requirements:

As we proceeded with the design, we focused on answering a series of question points defined in the prompt to fulfill the various design requirements.

1. Number of Operands :

Most instructions have **three** operands. For example, the instruction **z = x + y** has three operands in **z**, **x** and **y** respectively.

In our ISA, we specified 0-3 operands per instruction explicitly. Instructions with three operands are designed like -> [Destination, Source 01, Source 02] where each operand is 2 bits long.

2. Types of Operands :

We have used 3 types of operands in our design. They are mentioned below:

- Register based
- Memory based
- Mixed based

3. Number of Operations :

We have specified 16 operations for our 10-bit single cycle CPU.

We allocated 4 bits for the Opcode to represent our operations and 2 bits for each operand. This means there are 2^4 = 16 combinations. Since our instruction length is 10 bits long, we can incorporate 16 operations. [0000 - 1111]

4. Types of Operations :

We have 5 types of operations in our ISA design. They are listed below:

- ➔ **Arithmetic :** ADD, SUB, ADDi (3 Arithmetic Operations)
- ➔ **Logical :** AND, OR, ANDi, ORi, sll, srl (5 Logical Operations)
- ➔ **Conditional** :
  - ◆ slt, slti (2 Conditional Operations)
  - ◆ Beq, Bne (2 Conditional Branch Operations)
- ➔ **Unconditional Branch** : Jump -> J (1 Unconditional Branch Operations)
- ➔ **Data Transfer :** lw, sw

5. Formats :

We have chosen 3 formats for our ISA design. They are listed below with the number of bits allocated in each field :

- **R-Type**

Opcode	R_s (Source Register 01)	R_t (Source Register 02)	R_d (Destination Register)
4 bits	2 bits	2 bits	2 bits

- **I-Type**

Opcode	R_s (Source Register 01)	R_d (Destination Register)	Immediate
4 bits	2 bits	2 bits	2 bits

- **J-Type**

Opcode	Target Register Address
4 bits	6 bits

6. List of Instructions :

There are 16 instructions (and their types) we have chosen for our 10-bit single cycle CPU along with their respective OpCodes. They are classified according to their formats in the instruction set table.

7. Instruction Set Table :

The Instruction Set Table below shows all the instructions previously mentioned for our 10-bit single cycle CPU in detail. The format, type are mentioned along with examples and meaning for each instruction.

***Instruction Set Table* :**

Instruction	Type	Format	Example	Meaning	OpCode
ADD	**Arithmetic**	R	ADD $\$S_0$, $\$S_1$, $\$S_2$	$\$S_0 = \$S_1 + \$S_2$	0000
SUB	**Arithmetic**	R	SUB $\$S_0$, $\$S_1$, $\$S_2$	$\$S_0 = \$S_1 - \$S_2$	0001
AND	**Logic**	R	AND $\$S_0$, $\$S_1$, $\$S_2$	$\$S_0 = \$S_1 \cap \$S_2$	0010
ADDi	**Arithmetic**	I	ADDi $\$S_0$, $\$S_1$, 2	$\$S_0 = \$S_1 + 2$	0011
sll	**Logic**	I	sll $\$S_0$, $\$S_1$, 2	$\$S_0 = \$S_1 << 2$	0100
lw	**Data Transfer**	I	lw $\$S_0$, 4($\$S_1$)	$\$S_0 = Mem[\$S_1 + 4]$	0101
sw	**Data Transfer**	I	sw $\$S_0$, 4($\$S_1$)	$Mem[\$S_1 + 4] = \$S_0$	0110
J	**Unconditional Branch**	J	J 15	Branch at 15	0111

Instruction	Type	Format	Example	Meaning	

Beq	**Conditional Branch**	I	Beq S_0$, S_1$, 4	if(S_0$ == S_1$)
				{branch to instance at 4}
				else
				{continue to next instance}
					1000

slt	**Conditional**	R	slt S_0$, S_1$, S_2$	if(S_1$ < S_2$) then S_0$ = 1 else S_0$ = 0	1001
slti	**Conditional**	I	slti S_0$, S_1$, 2	if(S_1$ < 2) then S_0$ = 1 else S_0$ = 0	1010
IN	**Input**	J	IN S_0$	S_0$ = $BUFFER	1011
OUT	**Output**	J	OUT S_0$	$BUFFER = S_0$	1100

8. List of Registers :

We defined a total of 10 registers in our design for our 10-bit single cycle CPU. They are listed below with their name, number and usage explanation.

***Register Table* :**

Register Number	Name of the Registers	Usage
$0	**$$t_0$$**	Temporary data, not preserved by subprograms
$1	**$$S_0$$**	Saved registers, preserved by subprograms
$2	**$$S_1$$**	Saved registers, preserved by subprograms
$3	**$$S_2$$**	Saved registers, preserved by subprograms

9. Addressing Modes :

We have 6 addressing modes in our design. They are listed below:

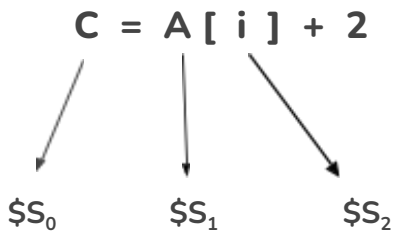
- Register addressing
- Immediate addressing
- Base addressing
- Indirect addressing
- Direct addressing
- PC-relative addressing

10. Examples :

For testing purposes, we have included an instruction from a high level language in each category from our objectives and tested our ISA design against them.

The three categories are classified and listed below:

- **Arithmetic and Logical Operations :**



***Compiled Code* :**

➔ sll S_2, S_2, 2

➔ ADD S_1, S_1, S_2

➔ lw t_0, 0(S_1)

➔ ADDi S_0, t_0, 2

- **Programs that require checking conditions :**

```
if (f == g)
        sum = f + g
else
        sum = f - g
```

Assume **f**, **g** and **sum** are in **S_0**, **S_1** and **S_2** respectively.

***Compiled Code* :**

- ➔ Bne $\$S_0$, $\$S_1$, L_1
- ➔ ADD $\$S_2$, $\$S_0$, $\$S_1$
- ➔ J L_2
- ➔ L_1 : SUB $\$S_2$, $\$S_0$, $\$S_1$
- ➔ L_2 : EXIT

- **Loop type of programs :**

for (i = 0 , i < 10 , i++)
A[i] = i * 2

Steps in the loop:

- ➢ i = 0 (initialization)
- ➢ i < 10 (check condition)
- ➢ A[i] = i * 2 (operation)
- ➢ i++ => i = i + 1 (increment i)
- ➢ Repeat condition check

Assume **i** and **A** are in **$\$S_0$** and **$\$S_1$** respectively.

- ❖ i → $\$S_0$
- ❖ A → $\$S_1$

***Compiled Code* :**

- ➔ SUB $\$S_0$, $\$S_0$, $\$S_0$ (i = 0)
- ➔ L_1 : slti $\$t_0$, $\$S_0$, 10 (i < 10 ; check)
- ➔ SUB $\$S_2$, $\$S_2$, $\$S_2$ (S_2 = 0 ; suppose)
- ➔ Beq $\$t_0$, $\$S_2$, L_2
- ➔ sll $\$t_0$, $\$S_0$, 1 (t_0 = [i x 2])
- ➔ sll $\$S_2$, $\$S_0$, 2 (S_2 = [i x 4])
- ➔ ADD $\$S_2$, $\$S_2$, $\$S_1$ ([i x 4] + S_1)
- ➔ sw $\$t_0$, 0($\$S_2$)
- ➔ ADDi $\$S_0$, $\$S_0$, 1
- ➔ J L_1
- ➔ L_2 : EXIT