EC5110 – Computer Architecture and Organization

Assignment 1 - Designing an ISA and a CPU

GROUP NO : 45

NAME : CHANDRASIRI H.V.B.L.

: GOWSIKAN N.

REGISTRATION NO. : 2019/E/023

: 2019/E/039

DATE : 12 NOV 2022

1. DESIGNING THE ISA

INSTRUCTION LENGTH

Here we choose the instruction length as 16 bits. It is not changing for the type of instruction. Only the instruction structure is changing with the type of instruction.

- R - Type

OP	RD	RA	RB
4 bits	4 bits	4 bits	4 bits

- I – Type

OP	RD	RA	INDEX
4 bits	4 bits	4 bits	4 bits

- J – Type

OP	JA
4 bits	12 bits

- K – Type

OP	RE	CON
4 bits	4 bits	8 bits

REGISTERS

We selected 2^4 (16) as the number of Register8s.

R1 – 0000	R5 - 0100	R9 – 1000	R13 - 1100
R2 – 0001	R6 – 0101	R10 – 1001	R14 - 1101
R3 – 0010	R7 – 0110	R11 – 1010	R15 – 1110
R4 – 0011	R8 – 0111	R12 – 1011	PTR – 1111

CAPABILITIES

We selected 2^4 (16) as the number of instructions.

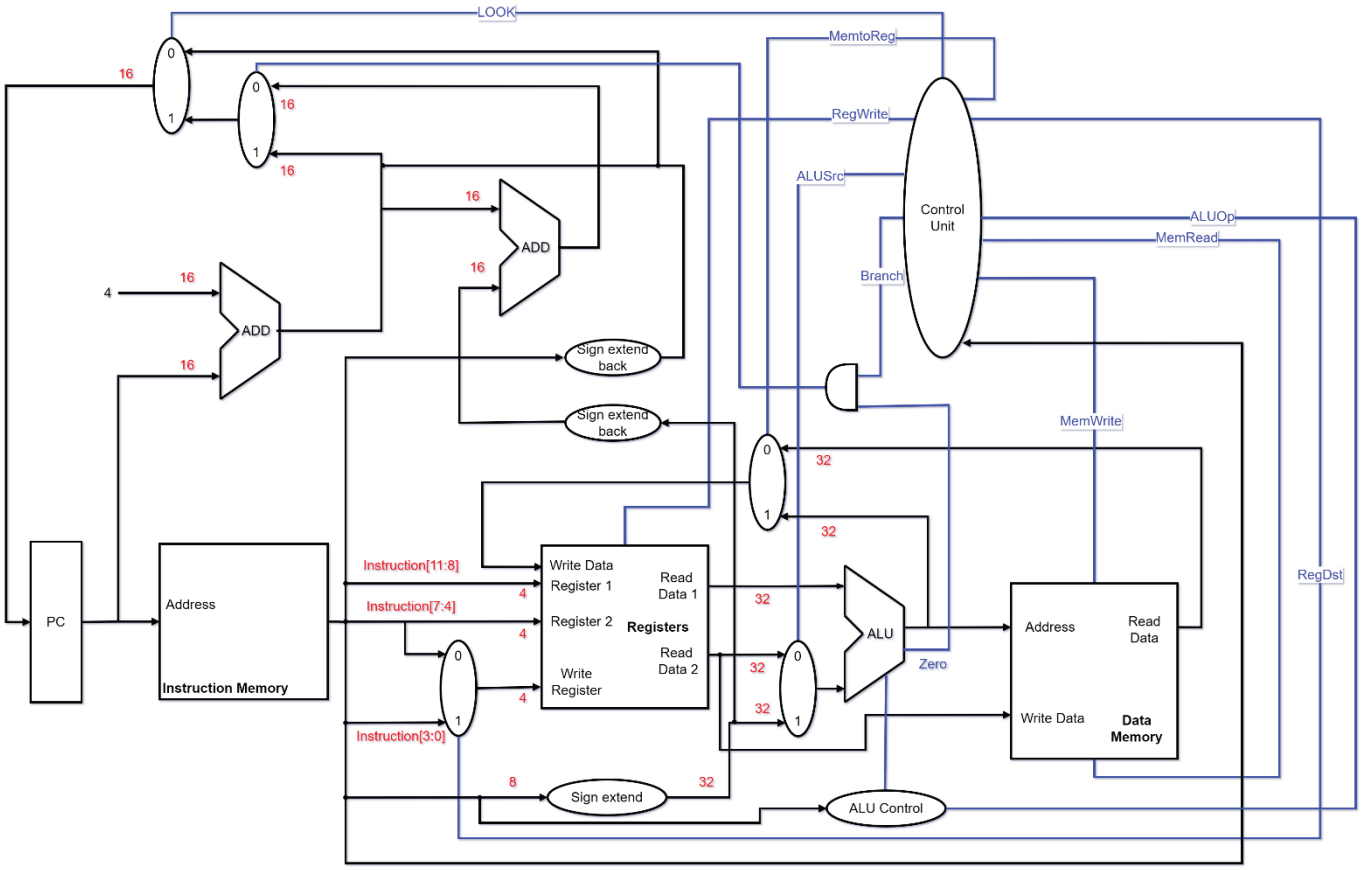
	Capability	*Instruction*	*Example*	*Description*
01	Data Movement	LCR	LCR $R1, 1	Load Constant to a Register
02		LCM	LCM 8($R9), 1	Load Constant to the memory
03		CRM	CRM $R1,8($R9)	Copy Register Value to Memory
04		CMR	CMR 8($R9), $R1	Copy Memory Value to Register
05	Logic	AND	AND $R1, $R2, $R3	Bitwise AND Operation
06		OR	OR $R1, $R2, $R3	Bitwise OR Operation
07		NOT	NOT $R1, $R2	Bitwise NOT Operation
		ADDONE	ADDONE $R1,$R2	Add one to register R2 = R1 + 1
08	Arithmetic	ADD	ADD $R1, $R2, $R3	Add First Two Values
09		SUB	SUB $R1, $R2, $R3	Subtract the Second Value from the First
10		MUL	MUL $R1, $R2, $R3	Multiply Two Values
11		DIV	DIV $R1, $R2, $R3	Divide the First Value by the Second Value
12	Unconditional Branching	LOOK	LOOK 100	Jump to the Given Address
13	Conditional Branching	COMPARE	COMPARE $R1, $R2, L	Compare given value go to label ($R1>R2)
14		EQUAL	EQUAL $R1, $R2, L	Compare equality of given value go to label

2. Designing a CPU to implement the ISA

INSTRUCTION ENCODING

Type	Function	Instruction			
R - Type	ADD	0000	RD	RA	RB
	SUB	0001	RD	RA	RB
	MUL	0010	RD	RA	RB
	DIV	0011	RD	RA	RB
	AND	0100	RD	RA	RB
	OR	0101	RD	RA	RB
I - Type	LCM	1000	RD	RA	INDEX
	CRM	1001	RD	RA	INDEX
	CMR	1010	RD	RA	INDEX
	COMPARE	1011	RD	RA	INDEX
	EQUAL	1100	RD	RA	INDEX
	ADDONE	1101	RD	RA	INDEX
K - Type	LCR	1110	RE	CONSTANT	
J - Type	LOOK	1111	JA		

CPU DESIGN



CONTROL LINES FOR THE CPU DESIGN

Control Line	R-Format Instruction	CMR	CRM	EQUAL	COMPARE
RegDst	1	0	0	0	0
ALUSrc	0	1	1	0	0
MemtoReg	0	1	1	0	0
RegWrite	1	1	0	0	0
MemRead	0	1	0	0	0
MemWrite	0	0	1	0	0
Branch	0	0	0	1	1
ALUOp1	1	0	0	0	0
ALUOp0	0	0	0	1	1

MEMORY

- We selected 2^5 (32) as the size of our Memory. The below tables show how 2 matrixes are saved in the memory.

Index	0	1	2	3	4	5	6	7	8
Value	A	B	C	D	E	F	G	H	I

Index	9	10	11	12	13	14	15	16	17
Value	J	K	L	M	N	O	P	Q	R

3. Write an assembly program

- We assume the Base address of Stack is saved in the $R16 register.
- Code is shown below.

```
# base address of the matrix 1 is $R15
# base address of the matrix 2 is $R16
# sum = $R14
# mul = $R13

LCR $R14,0
LCR $R13,1

LCR $R1,-4              # i = R1
LCR $R2,2

LCR $R3,0
LCR $R4,3
LCR $R5,1
LCR $R6,4

LCR $R9,-12             # j = R9
LCR $R10,12

LOOP1:
        EQUAL R3,R4,LOOP2
```

```
        ADD R1,R6,R7          # R7 = R6+R1 = 4 + i
        ADD R7,R15,R7                 # R7 = R7+R15
        CMR 0(R7),R8

        ADD R9,R10,R11                # R11 = R10+R9 = 12 + j
        ADD R11,R16,R11               # R11 = R11+R16
        CMR 0(R11),R12

        MUL R12,R8,R13
        ADD R14,R13,R14               # R14 = R14+R13

        ADDONE R3,R3                  # R3 = R3+1

        COMPARE R5,R4,LOOPEND
        LOOK LOOP1

LOOP2:
        LCR R1,-4
        LCR R9,-8
        LCR R3,0
        EQUAL R5,R2,LOOP3
        ADDONE R5,R5
        LOOK LOOP1

LOOP3:
        LCR R1,-4
        LCR R9,-4
        LCR R3,0
        EQUAL R5,R4,LOOPEND
        ADDONE R5,R5
        LOOK LOOP1

LOOPEND:

LCR $R14,0
LCR $R13,1

LCR $R1,8             # i = R1
LCR $R2,2

LCR $R3,0
LCR $R4,3
LCR $R5,1
LCR $R6,4

LCR $R9,-12           # j = R9
LCR $R10,12

LOOP4:
```

```
        EQUAL R3,R4,LOOP5

        ADD R1,R6,R7          # R7 = R6+R1 = 4 + i
        ADD R7,R15,R7                 # R7 = R7+R15
        CMR 0(R7),R8

        ADD R9,R10,R11                # R11 = R10+R9 = 12 + j
        ADD R11,R16,R11               # R11 = R11+R16
        CMR 0(R11),R12

        MUL R12,R8,R13
        ADD R14,R13,R14               # R14 = R14+R13

        ADDONE R3,R3                  # R3 = R3+1

        COMPARE R5,R4,LOOPEND2
        LOOK LOOP4

LOOP5:
        LCR R1,8
        LCR R9,-8
        LCR R3,0
        EQUAL R5,R2,LOOP6
        ADDONE R5,R5
        LOOK LOOP4

LOOP6:
        LCR R1,8
        LCR R9,-4
        LCR R3,0
        EQUAL R5,R4,LOOPEND2
        ADDONE R5,R5
        LOOK LOOP4

LOOPEND2:

LCR $R14,0
LCR $R13,1

LCR $R1,20            # i = R1
LCR $R2,2

LCR $R3,0
LCR $R4,3
LCR $R5,1
LCR $R6,4

LCR $R9,-12           # j = R9
LCR $R10,12
```

```
LOOP7:
        EQUAL R3,R4,LOOP8

        ADD R1,R6,R7          # R7 = R6+R1 = 4 + i
        ADD R7,R15,R7                 # R7 = R7+R15
        CMR 0(R7),R8

        ADD R9,R10,R11                # R11 = R10+R9 = 12 + j
        ADD R11,R16,R11               # R11 = R11+R16
        CMR 0(R11),R12

        MUL R12,R8,R13
        ADD R14,R13,R14               # R14 = R14+R13

        ADDONE R3,R3                  # R3 = R3+1

        COMPARE R5,R4,LOOPEND3
        LOOK LOOP7

LOOP8:
        LCR R1,8
        LCR R9,-8
        LCR R3,0
        EQUAL R5,R2,LOOP9
        ADDONE R5,R5
        LOOK LOOP7

LOOP9:
        LCR R1,8
        LCR R9,-4
        LCR R3,0
        EQUAL R5,R4,LOOPEND3
        ADDONE R5,R5
        LOOK LOOP7

LOOPEND3:

#AVERAGE
LRC $R1,9
DIV $R14,$R1,$R14


# R14 IS AVERAGE OF MULTIPLICATION OF THE MATRIX
```