North South University

Department of Electrical & Computer Engineering

CSE332

Computer Organization and Architecture

Instruction Set Architecture Design

Submitted by:

Name: Md. Arifur Rahaman

ID: 1831253642

Submitted to:

Tanjila Farah (TnF)

Objectives: My objective was to design a 13 Bit ISA which can solve a particular problems like simple arithmetic & logic operations, branching and loops.

Operands: My goal is to use accumulator base ISA. For this reason, I am going to take two operands. I will address these two operands as **d**, **t** and **s**.

Types of Operands: I need register operands to implement arithmetic instructions, and memory operands to implement data transfer instructions from memory to register. As a result, I'll need two sorts of operands.

- Register based.
- Memory based.

Operations: I will allocate 4 bits opcode, so the executable instructions number will be 2^4 or 16.

Types of operations: In my design there will be five different types of operation. The operations are:

- Arithmetic
- Logical
- Data Transfer
- Conditional Branch
- Unconditional Jump

Category	Operation	Name	Type	OpCode	Syntax	Comments
	No operation	nop		0000	nop	
Data transfer	Store word	sw	I	0001	sw r0 r1 2	Mem[r0+2] = r1
Arithmetic	Subtraction	sub	R	0010	sub r1 r2 r3	r3 = r1-r2
Conditional	Compare less than	slt	R	0011	slt r1 r2 r3	If(r1<r2) then r3 = 1 else r3 = 0
Conditional	Check equality	beq	I	0100	beq r1 r2 4	If(r1==r2) then go to line 4
Logical	Shift left	sll	R	0101	sll r1 r2 r3	r3 = r1<<r2
Unconditional	Jump	jmp	J	0110	jmp 6	Go to line 6
Arithmetic	Add two numbers	add	R	0111	add r1 r2 r3	r3 = r1 + r2
Data transfer	Load word	lw	I	1000	lw r0 r1 2	r1 = Mem[r0+2]
Arithmetic	Add number with an immediate	addi	I	1001	addi r1 r2 5	r2 = r1+5
Logical	Bit-by-bit and	and	R	1010	and r1 r2 r3	R3 = r1 & r2

Formats:

I use three types of formats for my ISA. They are:

- Register Type – R type
- Immediate Type – I type
- Jump Type - J Type

R Type ISA Format

OpCode	rs	rt	rd
4 bits	3 bits	3 bits	3 bits

I Type ISA Format

OpCode	rs	rt	Immediate
4 bits	3 bits	3 bits	3 bits

J Type ISA Format

OpCode	Target Address
4 bits	9 bits

List of Register:

As we have allocated four bits register so the number of registers will be $2^4 = 16$.

Register Number	Conventional Name	Usage	Binary Value
0	R0	General Purpose	0000
1	R1	General Purpose	0001
2	R2	General Purpose	0010
3	R3	General Purpose	0011
4	R4	General Purpose	0100
5	R5	General Purpose	0101
6	R6	General Purpose	0110
7	R7	General Purpose	0111

Translating some HLL codes using my designed 13-bit ISA

1. a = a + b # r1 = a, r2 = q

add r1 r2 r1 # r1 gets r1 + r2

2. a = a – b # r1 = a, r2 = b

sub r1 r2 r1 # r1 gets r1 – r2

3. a = a and b # r1 = a, r2 = b

and r1 r2 r1 # r1 gets r1 and r2

4. if(a<b) # r1 = a, r2 = b

c = 1 # r3 = c

slt r1 r2 r3 # r3 gets 1 if(r1<r2) else r3 gets 0

5.C = A[i] #r1 = I, r2=A , r3=C

sll r1, 2 # rp = r1<< 2

add r2 rp #r2 = r2 + rp

lsp r2 # rp = r2

lw r3 0 # getting the location value of A[i] from rp
#and store them in s3.

6. if(i ==4)
i = i+2
else

i = i -2 # r1 = i

sub rp, rp # rp = 0

addi rp, 4 # rp = 4

beq r1 L #comparing r1 = i with rp = 4. If equal
then go to L

addi r1, -2 # i = i – 2

jmp exit # Jump to exit

L: addi r1, 2 # i = i + 2

7. for(int j = 3; j < a; j++){
b = b*8;
} # r1 = j, r2 = a, r3 = b

sub r1, r1 # r1 = 0

addi r1, 3 # initializing r1 = 3

```
sub rp, rp                    # rp = 0

add rp, $s2                   # rp = r2

L2: beq r1 L1                 #if r1 = sp then go to L1, otherwise
continue loop.

sll r3 3                      # rp = r3 << 3

add r3 $sp                    # r3 = r3 * 8

addi r1 1                     # increment r1 by 1 / i++

jmp L2                        # go to the loop
```

Limitations: I have divided my 13 bits in 4 divisions. That's why we couldn't reserve space for shift amount. Thus, we used the immediate 3bit space in I type format for shifting purpose but it doesn't allow me to take the immediate value which need more then three bit binary value to represent itself at a time.