Junliang Fei, Adam Dubs, and Connelly Bale
Computer Architecture ECE 3570
Due Date: February 5, 2018

Lab One

This ISA is design for a 10-bit processor.

BASIC INSTRUCTION FORMATS

R-TYPE (opcode = 00 or 01)

opcode		function		rs			rt or immediate		
bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0

I-TYPE (opcode = 10)

opcode		immediate							
bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0

J-TYPE (opcode = 11)

opcode		function		address					
bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0

INSTRUCTION SET

No.	NAME	MNEMONIC	FORMAT	OPERATION	FORMAT	Example	OPCODE/ FUNCT
1	Add	add	R	**$t0** = $rs + $rt	add rs, rt	add $s0, $s1	00/00
2	Sub	sub	R	**$t0** = $rs - $rt	sub rs, rt	sub $s0, $s1	00/01
3	Compare	cmp	R	If($rs = $rt) **$t0** = 1	cmp rs, rt	cmp $s0, $s1	00/10
4	Less Than	slt	R	If($rs < $rt) **$t0** = 1	slt rs, rt	slt $s0, $s1	00/11
5	Move	mov	R	$rs =$rt	mov rs, rt	mov $s0, $t0	01/00
6	Load Word*	load	R	**$t1** =imm($rs)	load offset($rs)	load 4($sp)	01/01
7	Store Word*	str	R	imm($rs) = **$t1**	str offset($rs)	str 0($sp)	01/10
8	Return	ret	R	PC= $ra	ret	ret	01/11
9	load immediate**	ldi	I	**$t1** = immediate	ldi number	ldi 17	10
10	Jump	j	J	PC+= JumpAddr	j target	j LOOP	11/00
11	Jump true	je	J	If(**$t0**=True) PC+= JumpAddr	je target	je LOOP	11/01
12	Jump and link	jal	J	$ra= PC; PC+= JumpAddr	jal target	jal LOOP	11/10
13	Halt	halt	J	ends program	halt	halt	11/11

Junliang Fei, Adam Dubs, and Connelly Bale
Computer Architecture ECE 3570
Due Date: February 5, 2018

REGISTER LIST

No.	Name	Use	Bit	Notes
1	$s0	Saved Temporary 0	000	
2	$s1	Saved Temporary 1	001	
3	$s2	Saved Temporary 2	010	
4	$s3	Saved Temporary 3	011	
5	$t0	Temporary Result 0	100	$t0 is used for saving the return value of add, sub, slt and cmp
6	$t1	Temporary Result 1	101	$t1 is used for saving the return value of load, str and ldi
7	$ra	Return Address	110	
8	$sp	Stack Pointer	111	Start address is 1023 and grows down

The memory size is 1024 × 10bit. Memory is addressed 0-1023.

*Addressing Mode: ##(register) = ## + value in register

**Load immediate instruction (ldi) sign extends 8-bit immediate to 10-bit register $t1

Motivation for Design

We found the more registers we had in our ISA the less instructions we could have. Therefore, we debated on having only 4 registers so we could set two registers as the operands with a third register as a result. This would leave 4 bits for instructions. However, this caused too much use of the load and store instructions when writing the code. Therefore, we chose to have 8 registers. Operations needing the address to two registers in the instruction would have the result sent to a predetermined result register so we could still keep 4 bits for instructions. Using two different temporary result registers reduces the need to move data between registers as frequently.

The sorting program requires data to be entered into a register of 8-bit width. Therefore, one instruction was created that only had two bits reserved for the type of instruction with the other 8 bits reserved for entering an immediate.

Each of the other instructions was created based upon need.

Junliang Fei, Adam Dubs, and Connelly Bale
Computer Architecture ECE 3570
Due Date: February 5, 2018

Descending Bubble Sort

Array of size n is already in memory. 1<n<=256.

```
        ldi     n-1
        mov     $s0, $t1
        ldi     4
        sub     $sp, $t1
        ldi     base address value
        str     0($sp)
begin:
        ldi     0
        slt     $t1, $s0
        je      end
        mov     $s1, $t1
inner:
        slt     $s1, $s0
        je      ifst
        ldi     1
        sub     $s0, $t1
        mov     $s0, $t0
        j       begin
ifst:
        load    0($sp)
        add     $t1, $s1
        load    0($t0)
        mov     $s2, $t1
        load    1($t0)
        move    $s3, $t1
        slt     $s2, $s3
        je      swap
incj:
        ldi     1
        add     $s1, $t1
        move    $s1, $t0
        j inner
swap:
        load    0($sp)
        add     $t1, $s1
        mov     $t1, $s3
        str     0($t0)
        mov     $t1, $s2
        str     1($t0)
        j       incj
end:
        halt
```

Dynamic Instruction Count:
Assuming Worse Case (Array already sorted Ascending)
7 Instructions executed once
3 Instruction executed once more than n – 1 times
8 instructions executed n – 1 times
21 instructions executed
    (n-2)+(n-3)+…+2+1 times
    or (n-2)(n-1)/2 times

Total Dynamic Instruction Count:
7+3+8(n-1)+21*(n-2)(n-1)/2

Junliang Fei, Adam Dubs, and Connelly Bale
Computer Architecture ECE 3570
Due Date: February 5, 2018

f = x * y – 4

$s0 = y; $s1 = y; $s2 = f

```
        ldi     x
        mov     $s0, $t1
        ldi     y
        mov     $s1, $t1
        ldi     0
        cmp     $s0, $t1
        je      end
        cmp     $s1, $t1
        je      end
        slt     $t1, $s0
        je      positiveX
        sub     $t1, $s0
        mov     $s0, $t0
        slt     $t1, $s1
        je      positiveY1
        sub     $t1, $s1
        mov     $s1, $t0
        jal     mul
        j       end
positiveY1:
        jal     mul
        sub     $t1, $t2
        j       end
positiveX:
        slt     $t1, $s1
        je      positiveY2
        sub     $t1, $s1
        mov     $s1, $t0
        jal     mul
        sub     $t1, $s2
        j       end
positiveY2:
        jal mul
end:
        ldi     4
        sub     $s2, $t1
        mov     $s2, $t0
        halt
```

```
mul:
        ldi     0
        slt     $t1, $s0
        je      loop
        ret
loop:
        add     $s2, $s1
        mov     $s2, $t0
        ldi     1
        sub     $t1, $s0
        mov     $s0, $t0
        j       mul
```

Dynamic Instructions Count:
Worse Case: (both x & y negative)
23 Instructions executed once
4 Instructions executed once more than y times
10 Instruction executed y times

Total Dynamic Instruction Count:
23 + 4 + 10*y

Junliang Fei, Adam Dubs, and Connelly Bale
Computer Architecture ECE 3570
Due Date: February 5, 2018

Moving Array In Memory

This program assumes starting and target address is less than 256. More instructions must be added for address greater than 256. ISA requires addresses be less than 1024.

```
        ldi     1
        mov     $s3, $t1
        ldi     0
        mov     $s2, $t1
        ldi     Starting Base Address
        move    $s0, $t1
        ldi     Target Base Address
        move    $s1, $t1
begin:
        load    0($s0)
        str     0($s1)
        add     $s0, $s3
        mov     $s0, $t0
        add     $s1, $s3
        mov     $s1, $t0
        cmp     $t1, $s2
        je      end
        j       begin
end:
        ldi     10
        str     0($s1)
        halt
```

Machine Code

```
        1000000001
        0100011101
        1000000000
        0100010101
        10Starting Base Address
        0100001101
        10Target Base Address
        0100001101
        0101000000
        0110001000
        0000000011
        0100000100
        0000001011
        0100001100
        0010101010
        1101000001
        1100111000
        1000001010
        0110001000
        1111000000
```

Dynamic Instruction Count

11 Instructions Executed Once
8 Instructions Executed L times (lL= length of array)
1 Instructions executed L – 1 times

Total Dynamic Instruction Count
11+9L- 1