Bilkent University

Computer Engineering

CS 224 – Computer Organization

**Preliminary Design Report**

**Lab 3**

**Section 3**

**Gülsüm Güdükbay (21401148)**

**Date Of Lab: 08 March 2016**

**Assembly Language Equivalent Of The Machine Codes Given In The imem Module**

|  |  |  |  |
| --- | --- | --- | --- |
| Location (hex) | Binary | Machine Instruction (hex) | Assembly Language Equivalent |
| 0000 | 001000 00000 00010 0000000000000101 | 20020005 | addi $v0, $0, 5 |
| 0400 | 001000 00000 00011 0000000000001100 | 2003000c | addi $v1, $0, 12 |
| 0800 | 001000 00011 00111 1111111111110111 | 2067fff7 | addi $a3, $v1, -9 |
| 0c00 | 000000 00111 00010 00100 00000 100101 | 00e22025 | or $a0, $a3, 0($v0) |
| 1000 | 000000 00011 00100 00101 00000 100100 | 00642824 | and $a1, $v1, 0($a0) |
| 1400 | 000000 00101 00100 00101 00000 100000 | 00a42820 | add $a1, $a1, 0($a0) |
| 1800 | 000100 00101 00111 0000000000001010 | 10a7000a | beq $a1, $a3, 10 |
| 1c00 | 000000 00011 00100 00100 00000 101010 | 0064202a | slt $a0, $v1, 0($a0) |
| 2000 | 000100 00100 00000 0000000000000001 | 10800001 | beq $a0, $0, 1 |
| 2400 | 001000 00000 00101 0000000000000000 | 20050000 | addi $a1, $0, 0 |
| 2800 | 000000 00111 00010 00100 00000 101010 | 00e2202a | slt $a0, $a3, 0($v0) |
| 2c00 | 000000 00100 00101 00111 00000 100000 | 00853820 | add $a3, $a0, 0($a1) |
| 3000 | 000000 00111 00010 00111 00000 100010 | 00e23822 | sub $a3, $a3, 0($v0) |
| 3400 | 101011 00011 00111 0000000001000100 | ac670044 | sw $a3, 68($v1) |
| 3800 | 100011 00000 00010 0000000001010000 | 8c020050 | lw $v0, 80($0) |
| 3c00 | 000010 00000000000000000000010001 | 08000011 | j 17 |
| 4000 | 001000 00000 00010 0000000000000001 | 20020001 | addi $v0, $0, 1 |
| 4400 | 101011 00000 00010 0000000001010100 | ac020054 | sw $v0, 84($0) |
| 4800 | 000010 00000000000000000000010010 | 08000012 | j 18 |

**Register Transfer Level Expressions For The New Instructions**

**bge:**

IM[PC]

İf(RF[rs) >= RF[rt])

PC 🡨 PC+4+SignExt(imm)<<2

else

PC 🡨 PC+4

**jalr:**

IM[PC]

RF[rd] 🡨PC+4

PC🡨RF[rs]

**swapRM:**

IM[PC]

RF[rt] 🡨 DM[[RF[rs] + SignExt(imm)]

DM[RF[rs] + SignExt(imm)] 🡨 RF[rt]

PC 🡨 PC+4

**push:**

IM[PC]

RF[rs] 🡨 RF[rs] – 4 # decrement stack pointer first!

DM[RF[rs]-4] 🡨 RF[rt]

PC🡨PC+4

**Additions To The Datapath**

SignImm

CLK

A

RD

**Instruction**

**Memory**

+

4

A1

A3

WD3

RD2

RD1

WE3

A2

CLK

**Sign Extend**

**Register**

**File**

0

1

0

1

A

RD

**Data**

**Memory**

WD

WE

0

1

PC

0

1

PC'

Instr

25:21

20:16

15:0

5:0

SrcB

20:16

15:11

<<2

+

ALUResult

ReadData

WriteData

SrcA

PCPlus4

PCBranch

WriteReg

4:0

Result

31:26

RegDst

Branch

MemWrite

MemtoReg

ALUSrc

RegWrite

Op

Funct

**Control**

**Unit**

Zero

PCSrc

CLK

ALUControl

2:0

ALU

0

1

25:0

<<2

27:0

31:28

PCJump

Jump

1

0

BranchIfEqualOrGreater

1

0

jalr

0

1

1

0

-4

IsImmed

1

0

25:21

WriteToRs

Green: bge instruction

Pink: jalr instruction

Cyan: push instruction

**Main Control Table**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instr**  **uction** | **Op5:0** | **Reg**  **Write** | **Reg**  **Dst** | **Alu**  **Src** | **Bra**  **nch** | **Mem**  **Write** | **Mem**  **toReg** | **ALU**  **Op1:0** | **Ju**  **mp** | **Branch**  **EqualOr Greater** | **jalr** | **Is Immed** | **Write to rs** |
| R-type | 000000 | 1 | 1 | 0 | 0 | 0 | 0 | 10 | 0 | X | 0 | X | 0 |
| lw | 100011 | 1 | 0 | 1 | 0 | 0 | 1 | 00 | 0 | X | 0 | 1 | X |
| sw | 101011 | 0 | X | 1 | 0 | 1 | X | 00 | 0 | X | X | 1 | X |
| beq | 000100 | 0 | X | 0 | 1 | 0 | X | 01 | 0 | 0 | X | X | X |
| j | 000010 | 0 | X | X | X | 0 | X | XX | 1 | X | 0 | X | X |
| addi | 001000 | 1 | 0 | 1 | 0 | 0 | 0 | 00 | 0 | X | 0 | 1 | X |
| Bge | 010101 | 0 | X | 0 | 1 | 0 | X | 11 | 0 | 1 | X | X | X |
| Jalr | 101111 | 1 | 1 | X | X | 0 | X | XX | 1 | X | 1 | X | 0 |
| Push | 101100 | 1 | 1 | 1 | 0 | 1 | 0 | 00 | 0 | X | 0 | 0 | 1 |
| SwapRM | 111111 | 1 | 0 | 1 | 0 | 1 | 1 | 00 | 0 | X | 0 | 1 | X |

**MIPS Assembly Code**

#########################################

# #

# text segment #

# #

#########################################

.text

addi $s0, $0, 10

addi $s1, $0, 50

addi $s2, $0, 0

#add

add $s2, $s1, $s0 #10+50

la $a0,($s2) # print integer result

addi $v0, $0, 1 # using syscall 1

syscall

la $a0, endl

addi $v0, $0 ,4 # prints endl

syscall

#sub

sub $s2, $s2, $s1 #60-50

la $a0,($s2) # print integer result

addi $v0, $0, 1 # using syscall 1

syscall

la $a0, endl

addi $v0, $0 ,4 # prints endl

syscall

#or

addi $s0, $0, 1

addi $s1, $0, 0

addi $s2, $0, 0

or $s2, $s1, $s0 #1|0

la $a0,($s2) # print integer result

addi $v0, $0, 1 # using syscall 1

syscall

la $a0, endl

addi $v0, $0 ,4 # prints endl

syscall

#and

addi $s0, $0, 1

addi $s1, $0, 0

addi $s2, $0, 0

and $s2, $s1, $s0 #1&0

la $a0,($s2) # print integer result

addi $v0, $0, 1 # using syscall 1

syscall

la $a0, endl

addi $v0, $0 ,4 # prints endl

syscall

#slt

addi $s0, $0, 10

addi $s1, $0, 50

addi $s2, $0, 0

slt $s2, $s0, $s1 #s2 = (s0 < s1) 1:0

la $a0,($s2) # print integer result

addi $v0, $0, 1 # using syscall 1

syscall

la $a0, endl

addi $v0, $0 ,4 # prints didn't branch!

syscall

#lw

lw $s3, word

la $a0,($s3) # print integer result

addi $v0, $0, 1 # using syscall 1

syscall

la $a0, endl

addi $v0, $0 ,4 # prints endl

syscall

#sw

#sw $s3, 4($0) #s2 now stores 42

#la $a0,($s2) # print integer result

#addi $v0, $0, 1 # using syscall 1

#syscall

la $a0, endl

addi $v0, $0 ,4 # prints endl

syscall

#beq

beq $s0, $s1, branch\_equal

la $a0, didnt\_branch

addi $v0, $0 ,4 # prints didn't branch!

syscall

la $a0, endl

addi $v0, $0 ,4 # prints endl

syscall

#bge

slt $t0, $s0, $s1

beq $t0, $0, branch

la $a0, didnt\_branch

addi $v0, $0 ,4 # prints didn't branch!

syscall

la $a0, endl

addi $v0, $0 ,4 # prints endl

syscall

#jump

j statement

statement:

la $a0, jumped

addi $v0,$0, 4 # prints jumped here!

syscall

#jalr

addi $t3, $0, 4

addi $t4, $0, 0

la $t5, label

label:

add $ra, $t3 ,$t5

addi $t4, $t4, 1

j jalr\_label

jalr\_label:

beq $t4, 2, jalr\_end

jr $ra

jalr\_end:

#swapRM

addi $s0, $0, 50

lw $s1, memory\_element

la $a0, endl

addi $v0, $0 ,4 # prints endl

syscall

la $a0,($s0) # print integer result

addi $v0, $0, 1 # using syscall 1

syscall

la $a0, endl

addi $v0, $0 ,4 # prints endl

syscall

la $a0,($s1) # print integer result

addi $v0, $0, 1 # using syscall 1

syscall

la $a0, endl

addi $v0, $0 ,4 # prints endl

syscall

addi $t0, $s0, 0

addi $s0, $s1, 0

sw $t0, memory\_element

la $a0,($s0) # print integer result

addi $v0, $0, 1 # using syscall 1

syscall

la $a0, endl

addi $v0, $0 ,4 # prints endl

syscall

lw $s1, memory\_element

la $a0,($s1) # print integer result

addi $v0, $0, 1 # using syscall 1

syscall

la $a0, endl

addi $v0, $0 ,4 # prints endl

syscall

#push

addi $sp, $sp, -4

sw $s1, 0($sp)

branch:

branch\_equal:

addi $v0,$0, 10 #system call to exit

syscall # bye bye

#########################################

# #

# data segment #

# #

#########################################

.data

word: .word 42

didnt\_branch: .asciiz "Didn't branch!"

jumped: .asciiz "Jumped here!"

memory\_element: .word 99

endl: .asciiz "\n"

##