# Computation Structures

## **Instruction Set Architecture Worksheet**

### **Summary of β Instruction Formats**

#### **Operate Class:**

31	26	25	21	20	16	15	11	10		0
10x	XXXX	R	c	R	la	R	h		unused	

Register	Symbol	Usage
R31	R31	Always zero
R30	XP	Exception pointer
R29	SP	Stack pointer
R28	LP	Linkage pointer
R27	BP	Base of frame pointer

OP(Ra,Rb,Rc): Reg[

 $Reg[Rc] \leftarrow Reg[Ra]$  op Reg[Rb]

Opcodes: ADD (plus), SUB (minus), MUL (multiply), DIV (divided by)

AND (bitwise and), OR (bitwise or), XOR (bitwise exclusive or), XNOR (bitwise exclusive nor), CMPEQ (equal), CMPLT (less than), CMPLE (less than or equal) [result = 1 if true, 0 if false] SHL (left shift), SHR (right shift w/o sign extension), SRA (right shift w/ sign extension)

31	26	25 21	20 16	15	0
	11xxxx	Rc	Ra	literal (two's complement)	

OPC(Ra, literal, Rc):  $Reg[Rc] \leftarrow Reg[Ra]$  op SEXT(literal)

Opcodes: ADDC (plus), SUBC (minus), MULC (multiply), DIVC (divided by)

ANDC (bitwise and), ORC (bitwise or), XORC (bitwise exclusive or), XNORC (bitwise exclusive nor) CMPEQC (equal), CMPLTC (less than), CMPLEC (less than or equal) [result = 1 if true, 0 if false] SHLC (left shift), SHRC (right shift w/o sign extension), SRAC (right shift w/ sign extension)

#### Other:

31	26	25	1 2	20 16	15	0
	01xxxx	Rc		Ra	literal (two's complement)	

LD(Ra,literal,Rc): $Reg[Rc] \leftarrow Mem[Reg[Ra] + SEXT(literal)]$ ST(Rc,literal,Ra): $Mem[Reg[Ra] + SEXT(literal)] \leftarrow Reg[Rc]$ JMP(Ra,Rc): $Reg[Rc] \leftarrow PC + 4$ ;  $PC \leftarrow Reg[Ra]$ 

**BEQ/BF**(Ra,label,Rc): Reg[Rc]  $\leftarrow$  PC + 4; if Reg[Ra] = 0 then PC  $\leftarrow$  PC + 4 + 4\*SEXT(literal) **BNE/BT**(Ra,label,Rc): Reg[Rc]  $\leftarrow$  PC + 4; if Reg[Ra]  $\neq$  0 then PC  $\leftarrow$  PC + 4 + 4\*SEXT(literal)

**LDR**(label,Rc):  $Reg[Rc] \leftarrow Mem[PC + 4 + 4*SEXT(literal)]$ 

#### **Opcode Table: (\*optional opcodes)**

2:0								
5:3	000	001	010	011	100	101	110	111
000								
001								
010								
011	LD	ST		JMP	BEQ	BNE		LDR
100	ADD	SUB	MUL*	DIV*	CMPEQ	CMPLT	CMPLE	
101	AND	OR	XOR	XNOR	SHL	SHR	SRA	
110	ADDC	SUBC	MULC*	DIVC*	CMPEQC	CMPLTC	CMPLEC	
111	ANDC	ORC	XORC	XNORC	SHLC	SHRC	SRAC	

#### Problem 1.

An unnamed associate of yours has broken into the computer (a Beta of course!) that 6.004 uses for course administration. He has managed to grab the contents of the memory locations he believes holds the Beta code responsible for checking access passwords and would like you to help discover how the password code works. The memory contents are shown in the table below:

Addr	Contents	Opcode	Rc	Ra	Rb	Assembly
0x100	0xC05F0008	110000	00010	11111		
0x104	0xC03F0000	110000	00001	11111		
0x108	0xE060000F	111000	00011	00000		
0x10C	0xF0210004	111100	00001	00001		
0x110	0xA4230800	101001	00001	00011		
0x114	0xF4000004	111101	00000	00000		
0x118	0xC4420001	110001	00010	00010		
0x11C	0x73E20002	011100	11111	00010		
0x120	0x73FFFFF9	011100	11111	11111		
0x124	0xA4230800	101001	00001	00011		
0x128	0x605F0124	011000	00010	11111		
0x12C	0x90211000	100100	00001	00001		

Further investigation reveals that the password is just a 32-bit integer which is in R0 when the code above is executed and that the system will grant access if R1 = 1 after the code has been executed. What "passnumber" will gain entry to the system?

<b>Problem</b>	2.
----------------	----

		-	ould a compiler use to impler available? Assume x is in I	ment $y = x * 8$ on the Beta assuming R0 and y is in R1.
			<b>Equivalent</b> as	sembly instruction:
	execution value of	on of each of the follo	wing assembly instructions. or memory location. If you	R2=12, R3=0x1234, R4=24 before For each instruction, provide the ar answers are in hexadecimal,
	1.	SHL(R3, R4, R5)	Value of R5:	_
	2.	ADD(R2, R1, R6)	Value of R6:	_
	3.	ADD(R0, 2, R7)	Value of R7:	
	4.	ST(R1, 4, R3)	Value stored:	at address:
	by A	ADDC(R0, 3*4+5, R: ADDC(R0, 17, R1)	am smaller? Does it run faste	er? ram is SMALLER? yes no
			(cir	rcle one) FASTER? yes no
(D)	that BR			. If the binary representation for ere, where will the relocated
			Branch target for reloca	ated BR (in hex): 0x
(E)				DDC(R1,2,R3)" is changed to e differently when executed?
			Circle best answer:	YES NO CAN'T TELL

#### Problem 3.

Each of the following programs is loaded into a Beta's main memory starting at location 0 and execution is started with the Beta's PC set to 0. Assume that all registers have been initialized to 0 before execution begins. Please determine the specified values after execution reaches the HALT() instruction and the Beta stops. Write "CAN'T TELL" if the value cannot be determined. Please write all values in hex.

(A) X:	LD(R31,X+4,R1) SHLC(R1,2,R1) LD(R1,X,R2) HALT() LONG(4) LONG(3) LONG(2) LONG(1) LONG(0)	Value left in R1: 0xValue left in R2: 0x
(B) L: DONE:	<pre>. = 0 LD(R31,X,R0) CMOVE(0,R1) CMPLTC(R0,0,R2) BNE(R2,DONE) ADDC(R1,1,R1) SHLC(R0,1,R0) BR(L) HALT() LONG(0x08306352)</pre>	Value left in R0: 0x  Value left in R1: 0x  Value left in R2: 0x  Value assembler assigns to symbol X: 0x
(C) Z:	<pre>. = 0 LD(R31,Z,R1) SHRC(R1,26,R1) CMPLTC(R1,0x3C,R2) HALT()</pre>	Value left in R1: 0x  Value left in R2: 0x
(D) L:	<pre>. = 0 LD(R31,X,R0) CMOVE(0,R1) ADDC(R1,1,R1) SHRC(R0,1,R0) BNE(R0,L,R2) HALT()</pre>	Value left in R0: 0xValue left in R1: 0xValue left in R2: 0x
X:	. = 0x100 LONG(5)	Value assembler assigns to symbol X: 0x

(E)		. = 0 LD(r31, X, r0) CMPLE(r0, r31, r1	) Value left in R0? 0x
	L1:	BNE(r1, L1, r1) ADDC(r31, 17, r2) BEQ(r31, L2, r31) SRAC(r0, 4, r2)	Value left in R1? 0x
		HALT()	Value left in R2? 0x
	x:	. = 0x1CE8 LONG(0x87654321)	Value assembler assigns to L1: 0x
(F)		. = 0	Contents of R0 (in hex): 0x
		LD(R31, i, R0) SHLC(R0, 2, R0) LD(R0, a-4, R1)	Contents of R1 (in hex): 0x
	a:	HALT() LONG(0xBADBABE)	
		LONG(0xDEADBEEF)	
		LONG(0xC0FFEE)	
		LONG(0x8BADF00D)	
	i:	LONG(3)	
(G)	L	= 0 D(R31,Z,R1)	Value left in R1: 0x
	Z: Sl	HRC(R1,16,R2) JBC(R2,0x3C,R3) ALT()	Value left in R3: 0x
			Value assembler assigns to symbol Z: 0x
(H)	L	= 0 D(R31,X,R0) MOVE(0,R1)	Value left in R0: 0x
		DDC(R1,1,R1)	Value left in R1: 0x
	BN	HRC(R0,1,R0) NE(R0,L,R2) ALT()	Value left in R2: 0x
	X: L0	ONG(0xDECAF)	

MIT OpenCourseWare <a href="https://ocw.mit.edu/">https://ocw.mit.edu/</a>

6.004 Computation Structures Spring 2017

For information about citing these materials or our Terms of Use, visit: <a href="https://ocw.mit.edu/terms">https://ocw.mit.edu/terms</a>.