Aidan Lambrecht
Dr. Harris
ELEC 5200
September 20, 2019

CPU Project Part One: ISA

Registers:

Register Name	Number/Address	Decription
$zero	0 / 0000	The Constant Value 0
$pc	1 / 0001	Program Counter
$ra	2 / 0010	Return Address
$flag	3 / 0011	For conditional operators
$temp0	4 / 0100	Temporary Register
$temp1	5 / 0101	Temporary Register
$temp2	6 / 0110	Temporary Register
$temp3	7 / 0111	Temporary Register
$arg0	8 / 1000	Arguments/Results
$arg1	9 / 1001	Arguments/Results
$arg2	10 / 1010	Arguments/Results
$arg3	11 / 1011	Arguments/Results
$s0	12 / 1100	Saved Registers
$s1	13 / 1101	Saved Registers
$s2	14 / 1110	Saved Registers
$s3	15 / 1111	Saved Registers

Instruction Formats:

R-Type: Used for mathematical operations between registers

Bits 15-12	Bits 11-8	Bits 7-4	Bits 3-0
0000	0000	0000	0000
Opcode	Destination Register	Operand Register 1	Operand Register 2

I-Type: Used for operations involving immediates

Bits 15-12	Bits 11-8	Bits 7-0
0000	0000	00000000
Opcode	Destination Register	Immediate Operand

L-Type: Used to load and store data between registers and memory

Bits 15-12	Bits 11-8	Bits 7-0
0000	0000	00000000
Opcode	Destination Register	Memory Location

J-Type: Used only for the JUMP command to address any space in the 1k word-addressed memory

Bits 15-12	Bits 9-0	Bits 11-10
0000	0000000000	00
Opcode	Memory Location	Unused

B-Type: Used for branch instructions

Bits 15-12	Bits 11-8	Bits 7-4	Bits 3-0
0000	0000	0000	0000
Opcode	Memory Location	Operand Register 1	Operand Register 2

I am unsure if similar formats like R- and B-Type can be considered the same format - the only difference between the two being the label on Bits 11-8. I have elected to leave both these two and I- and L-Types listed here, just in case they are needed.

Instructions:

Opcode	**Instruction**	**Description**	**Type**
0000	HALT	Halts operation	J
0001	LOAD	Loads word from memory, stores in register	L
0010	STOR	Stores word from register to memory	L
0011	ADDR	Adds values in two registers, stores in third	R
0100	ADDI	Adds value in a register and an immediate and stores in same register	I
0101	AND	Ands two registers together, stores result in third	R
0110	OR	Ors two registers together, stores result in third	R
0111	SHL	Shifts register contents left one bit, stores in same register	I
1000	SHR	Shifts register contents right one bit, stores in same register	I
1001	JUMP	Jumps to a specified location in memory (word-accessible)	J
1010	BRE	Branch to a memory address if two registers are equal	B
1011	BRNE	Branch to a memory address if two registers are not equal	B
1100	BRLT	Branch to a memory address if a register is less than another	B
1101	BRGE	Branch to a memory address if a register is greater than or equal to another	B
1110	SUBR	Subtracts values in two registers, stores in third	R
1111	XOR	Xors two registers together, stores result in third	R

Justification:

Instruction	**Description**
HALT	Required
LOAD	Necessary in order to interact with memory
STOR	Necessary in order to interact with memory
ADDR	Required
ADDI	Provides way to set registers (useful for constants and others)
AND	Required
OR	Required

SHL	Convenient multiplication capability
SHR	Convenient dividing capability
JUMP	Required/Provides addressing to 1k words of memory
BRE	Required/Provides logical operator capabilities (not actually necessary, but is convenient)
BRNE	Required/Provides logical operator capabilities
BRLT	Required/Provides logical operator capabilities
BRGE	Required/Provides logical operator capabilities
SUBR	Required
XOR	Convenient logical operator

Instruction Translations:

Instruction Name	**Assembly Language**	**Machine Language**
HALT	halt	0000 000000000000
LOAD	load $s0, 128	0001 1100 10000000
STOR	stor $s1, 128	0010 1101 10000000
ADDR	addr $s2, $temp0, $temp1	0011 1110 0100 0101
ADDI	addi $s2, 128	0100 1110 10000000
AND	and $s2, $arg0, $arg1	0101 1110 1000 1001
OR	or $s2, $arg2, $arg3	0110 1110 1010 1011
SHL	shl $s2	0111 1110 00000000
SHR	shr $s2	1000 1110 00000000
JUMP	jump label	1001 001000000000
BRE	bre label, $s2, $s1	1010 0111 1110 1101
BRNE	brne label, $s2, $s1	1011 0111 1110 1101
BRLT	brlt label, $s2, $s1	1100 0111 1110 1101
BRGE	brge label, $s2, $s1	1101 0111 1110 1101
SUBR	subr $s2, $temp0, $temp1	1110 0111 0100 0101
XOR	xor $s2, $temp0, $temp1	1111 0111 0100 0101

C Constructs:

Construct	**C Code**	**Assembly**
Add	x = a + b;	addr $s2, $temp0, $temp1
Subtract	x = a - b;	subr $s2, $temp0, $temp1
And	x = a & b;	and $s2, $temp0, $temp1
Or	x = a | b;	or $s2, $temp0, $temp1

If-else	if (a == b) {
	  x = a + b;
	}
		bre label, $temp0, $temp1
		jump done
		label: addr $s2, $temp0, $temp1
		done: halt

While	while (a == b) {
	  x = x + a;
	}
		label: brne done, $temp0, $temp1
		addr $s2, $temp0, $s2
		jump label
		done: halt

For	for (x = 0; x < 1; x++) {
	  x = x + a;
	}
		addi $temp0, 1
		label: brge done, $s2, $temp0
		addr $s2, $s2, $temp0
		jump label
		done: halt

Function Call (by value)	foo(x);	jump foo #ra is loaded with pc . . . foo: … #copy $s2 (x) into $temp0 and complete function with that value jump $ra
Function Call (by reference)	foo(&x);	jump foo #ra is loaded with pc . . . foo: … #use $s2 (the register holding the value x) jump $ra