Courtney Stewart
ELEC5200
CPU Design Project 1

Registers:

Register Number	Usage
R0	Constant zero value
R1-R15	Any use register

Instruction formats:

Type	Bits 15:12	Bits 11:8	Bits 7:4	Bits 3:0
R (arithmetic)	Source register 2	Source register 1	Destination register	Opcode
B (branch)	Destination address	Source register	R0	Opcode
I (immediate needed)	Immediate[7:4]	Immediate[3:0]	Destination register	Opcode
M (memory dealt with)	Destination address	Source register	R0	Opcode

Mnemonic	Type	Name	Description	Opcode	Justification
stor	M	store word	store word from register to memory address	0000	Need to store value to memory
load	M	load	load value from memory to register	0001	Need to load value from memory
lui	I	load upper immediate	load upper immediate value from memory	0010	Not enough bits to access all memory addresses, to use for upper half
lli	I	load lower immediate	load lower immediate value from memory	0011	Not enough bits to access all memory addresses, to use for lower half
add	R	add	R[Rd] = R[Rs1] + R[Rs2]	0100	Essential function
sub	R	subtract	R[Rd] = R[Rs1] - R[Rs2]	0101	Essential function
andl	R	logical and	R[Rd] = R[Rs1] & R[Rs2]	0110	Essential function
orl	R	logical or	R[Rd] = R[Rs1] | R[Rs2]	0111	Essential function
xor	R	logical xor	R[Rd] = R[Rs1] ^ R[Rs2]	1000	Useful for flipping all bits
lsl	R	logical shift left	R[Rd] = R[Rs1] << R[Rs2]	1001	For multiplying a value by 2^x
lsr	R	logical shift right	R[Rd] = R[Rs1] >> R[Rs2]	1010	For dividing a value by 2^x (not keeping sign)
asr	R	arithmetic shift right	R[Rd] = R[Rs1] >> R[Rs2]	1011	For dividing a value by 2^x (keeping sign)
beq	B	branch if equal to zero	if R[Rs] == R[R0] PC = R[Destination address]	1100	Check if two numbers are equal
blt	B	branch if less than zero	if R[Rs1] < R[R0] PC = R[Destination address]	1101	Only need three out of the full six operands to check them all
bge	B	branch if greater than zero	if R[Rs1] > R[R0] PC = R[Destination address]	1110	Only need three out of the full six operands to check them all
stop	R	stop/halt	PC = PC - 1	1111	Essential function

C construct	C code	Assembly language implementation
variable = expression	a = b + c	R2 = a, R3 = B, R4 = C add R2, R3, R4
addition	a = b + c	R2 = a, R3 = B, R4 = C add R2, R3, R4
subtraction	a = b - c	R2 = a, R3 = B, R4 = C sub R2, R3, R4
logical AND	a = a & b	R2 = a, R3 = b andl R2, R2, R3
logical OR	a = a l b	R2 = a, R3 = b orl R2, R2, R3
if-else statement	if (a == b) … else (a = 0)	R8 = a, R9 = b beq R8, R9, Label andl R8, R8, R0
while loops	while (a < b) …	R8 = a, R9 = b blt R8, R9, Label
==	a == b	R8 = a, R9 = b beq R8, R9, Equal
!=	a != b	R8 = a, R9 = b sub R7, R8, R9 beq R7, R0, notEqual

>	a > b	//check if b < a
		R8 = a, R9 = b
		blt R9, R8, aGreaterThan

<=	a <= b	//check if b >= a
		R8 = a, R9 = b
		bge R9, R8, bGreaterEqual

<	a < b	R8 = a, R9 = b blt R8, R9, aLessThan
>=	a >= b	R8 = a, R9 = b bge R8, R9, aGreaterEqual

Call and return functions	if (a ==b){
	add()
	}
	
	c = c - 1
	
	add(
	c = a+b;
	)
		R8 = a, R9 = b, R10 = c, R2 = 1
		beq R8, R9, equal
		sub R10, R10, R2
		
		equal
		add R10, R8, R9