Computer Organization and Architecture Laboratory

Verilog Assignment 7

32-bit RISC like Processor

GROUP-18

Sreejita Saha 21CS30052

Ratan Junior 21CS30041

Instruction Format Encoding

1. Number of General Purpose Registers: 16 (R0 - R15)
    - R0 is specifically used for zero value computation
2. Number of Special Purpose Registers: 2 (PC and SP)
3. Total number of registers: 18 (16 + 2)
4. Bits for addressing registers: 5
    - thus, a maximum of $2^5 = 32$ registers can be addressed

Types of Instruction Encoding

1. Common functions

Arithmetic, Logical and Shift Operations					
OPCODE	rs	rt	rd	shamt	func
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

First 6 bits will be reserved for opcode - [31:26]
Next 5 bits will be reserved for first source register - [25:21]
Next 5 bits will be reserved for second source register - [20:16]
Next 5 bits will be reserved for destination register - [15:11]
Next 5 bits will be reserved for shamt(shift value) - [10:6]
Last 6 bits will be reserved for func code - [5:0]

2. Immediate functions

Immediate Operations			
OPCODE	rs	rt	Immediate Data
6 bits	5 bits	5 bits	16 bits

First 6 bits will be reserved for opcode - [31:26]

Next 5 bits will be reserved for source register - [25:21]
Next 5 bits will be reserved for destination register - [20:16]
Last 16 bits will be reserved for immediate data - [15:0]

3. Branch and call functions

Branch and Call operations			
OPCODE	rs	Immediate Data	func
6 bits	5 bits	15 bits	6 bits

First 6 bits will be reserved for opcode - [31:26]
Next 5 bits will be reserved for a register - [25:21]
Next 15 bits will be reserved for immediate data i.e the destination address - [20:6]
Last 6 bits will be reserved for func code - [5:0]

Opcode Specification

Arithmetic, Logical and Shift Operations

- uses type 1 instruction encoding

Operations	**OPCODE** (in decimal)	**Function Code** (in decimal)
ADD	0	0
SUB	0	1
AND	1	0
OR	1	1
XOR	1	2
NOT	2	0

Operations	**OPCODE** (in decimal)	**Function Code** (in decimal)
SLA	3	0
SRA	3	1
SRL	3	2

Immediate Operations

- uses type 2 instruction encoding

Operations	**OPCODE** (in decimal)
ADDI	4
SUBI	5
ANDI	6
ORI	7
XORI	8
NOTI	9
SLAI	10
SRAI	11
SRLI	12
LD	13
ST	14
LDSP	15
STSP	16
MOVE	17
PUSH	18

Operations	**OPCODE** (in decimal)
POP	19

Branch and Call Operations

- uses type 3 instruction encoding

Operations	**OPCODE** (in decimal)	**Function Code** (in decimal)
BR	20	0
BMI	21	0
BPL	21	1
BZ	21	2
CALL	22	0
RET	23	0
HALT	24	0
NOP	25	0

Control Unit Signals

Opcode, func	Read reg	ALUop	MUXA1	MUXA2	WReg	MUXWB	RW Mem	Load SP	MUXMem Write	branch	desc
0,0	11	0000	0	0	1	0	00	00	x	000	add
0,1	11	0001	0	0	1	0	00	00	x	000	sub
1,0	11	0101	0	0	1	0	00	00	x	000	and
1,1	11	0111	0	0	1	0	00	00	x	000	or

Opcode, func	Read reg	ALUop	MUXA1	MUXA2	WReg	MUXWB	RW Mem	Load SP	MUXMem Write	branch	desc
1,2	11	1000	0	0	1	0	00	00	x	000	xor
2,0	10	0110	0	x	1	0	00	00	x	000	not
3,0	10	0011	0	x	1	0	00	00	x	000	sla
3,1	10	1001	0	x	1	0	00	00	x	000	sra
3,2	10	0100	0	x	1	0	00	00	x	000	srl
4	10	0000	0	1	1	0	00	00	x	000	addi
5	10	0001	0	1	1	0	00	00	x	000	subi
6	10	0101	0	1	1	0	00	00	x	000	andi
7	10	0111	0	1	1	0	00	00	x	000	ori
8	10	1000	0	1	1	0	00	00	x	000	xori
9	00	0110	x	1	1	0	00	00	x	000	noti
10	10	0011	0	1	1	0	00	00	x	000	slai
11	10	1001	0	1	1	0	00	00	x	000	srai
12	10	0100	0	1	1	0	00	00	x	000	srli
13	10	0000	0	1	1	1	10	00	x	000	ld
14	11	0000	0	1	0	x	01	00	0	000	st
15	10	0000	0	1	0	x	10	11	x	000	ldsp

Opcode, func	Read reg	ALUop	MUXA1	MUXA2	WReg	MUXWB	RW Mem	Load SP	MUXMem Write	branch	desc
16	10	0000	0	1	0	x	01	00	1	000	stsp
17	11	0000	0	0	1	0	00	00	x	000	move
18	01	xxxx	1	x	0	x	01	10	0	000	push
19	00	0000	1	1	1	1	10	01	x	000	pop
20,0	00	xxxx	x	x	0	x	00	00	x	001	br
21,0/1/2	10	0010	0	x	0	x	00	00	x	010	bmi, bpl, bz
22,0	00	xxxx	1	x	0	x	01	10	1	011	call
23,0	00	0000	1	1	0	x	10	01	x	100	ret
24,0	00	xxxx	x	x	0	x	00	00	x	101	halt
25,0	00	xxxx	x	x	0	x	00	00	x	000	nop

Notes:

- The Control Unit produces the signals by depending on the opcode and optionally func.
- **Readreg[1:0]** specifies which of the two read ports from the register bank are active, and is 00 and 11 if none or both are active respectively.
- **ALUop [3:0]** specifies the ALU operations and is x for instruction where ALU is not used or is disabled

- **MUXA1** selects read port A if 0 otherwise SP and **MUXA2** selects read port B if 0 otherwise sign-extended immediate field from type 2 instruction.
- **WReg** is 1 when there is a write operation to one of the registers in the bank otherwise it is 0. It also acts as an enable pin for **MUXWB**.
- **MUXWB** when enabled by WReg selects ALU output to be written in the reg bank when it is 0 otherwise (when = 1) selects the data read from the data memory.
- **RWMem [1:0]** specifies the read/write for the data memory, i.e. read is executed if MSB is 1 and otherwise not and write is executed if LSB is 1.
- **LoadSP [1:0]** is 2-bit control lines for 4-to-1 MUX for updating the SP, i.e. it takes on
    1. 00 for no update to SP
    2. 01 for SP = SP + 4
    3. 10 for SP = SP - 4
    4. 11 for SP being loaded
- **MUXMemWrite** selects read port B to be written into data memory if it is equal to 1 otherwise one of *SP MUX* output port or *PC+4* is written into the data memory at the specified memory address.
- **Branch [2:0]** is 3-bit control signal for branching mechanism related operations
- **MUX(SP/PC)** chooses *SP MUX* output port or *PC + 4* from the branching mechanism circuit depending on the **logical AND of branch[0] and branch[1] (branch[0] ^ branch[1)** taking on value 0 and 1 respectively.
- Move instruction reads R0 in port B and source register in port A and adds the two and the alu output is then stored in the destination register.
- For branch operations like BMI,BPL,BZ , we check in the ALU if the source data is zero, positive or negative and accordingly flag is assigned 1xx, x1x,xx1 values.

SCHEMATIC DIAGRAM: