INSTRUCTION FORMAT FOR THE PROCESSOR

GROUP- 53

21CS30017 – DEEPRAJ DAS

21CS30026 – JAY PRIYADARSHI

3 types of instructions:

1. R-type
2. I-type
3. J-type

Registers:

1. 16 x 32-bit General Purpose Registers R0-R15
2. Special Purpose registers:
   a. PC : Program Counter
   b. SP: Stack Pointer

Instruction Set Architecture Format

General R-type Instructions

opcode	Source register 1(rs)	Source Register 2(rt)	Destination Register (rd)	Extented opcode (function)
6 bits	5 bits	5 bits	5 bits	11 bits

I-type Instructions:

Opcode	R1 (source)	R2 (destination)	Immediate Data
6 bits	5 bits	5 bits	16 bits

Note: The register encoding requires only 4 bits but we are using 5 bits where the MSB will be a padded 0 value, so R15 will be 01111 instead of 1111, this will also help to perform ALU functions on special purpose registers without using extra opcodes for those.

ALU instructions and their encoding:

(R type addressing)

Instruction	Opcode (6)	R1 (5)	R2 (5)	R3 (5)	Extended opcode (11)
ADD	000000	rs	rt	rd	00…….0000
SUB	000000	rs	rt	rd	00…….0001
AND	000000	rs	rt	rd	00…….0010
OR	000000	rs	rt	rd	00…….0011
XOR	000000	rs	rt	rd	00…….0100
NOT	000000	rs	rt	rd	00…….0101
SLA	000000	rs	rt	rd	00…….0110
SRA	000000	rs	rt	rd	00…….0111
SRL	000000	rs	rt	rd	00…….1000

ALU I-type instructions and their encoding:

Instruction	opcode	R1 (source)	R2 (destination)	Immediate
ADDI	000001	rs	rt	16 bit val
SUBI	000010	rs	rt	16 bit val
ANDI	000011	rs	rt	16 bit val
ORI	000100	rs	rt	16 bit val
XORI	000101	rs	rt	16 bit val
NOTI	000110	rs	rt	16 bit val
SLAI	000111	rs	rt	16 bit val
SRAI	001000	rs	rt	16 bit val
SRLI	001001	rs	rt	16 bit val

In the case of shift operations, the immediate input can be either 0 or 1, if it is non zero we will shift it by 1.

Note: The register encoding requires only 4 bits but we are using 5 bits where the MSB will be a padded 0 value, so R15 will be 01111 instead of 1111

Load and Store instructions:

These will need I-type instructions

Instruction	opcode	R1	R2	Immediate value
LD/LDSP	001010	rs (address)	rt/SP (register)	16-bit offset value
ST/STSP	001011	rs (address)	rt/SP (register)	16 bit offset value

Note: Since we have used 5 bits for register indexing, we do not need different opcodes for LDSP and STSP instead we can encode SP using 5 bits register indexing and using the same opcode to perform similar function as in LD and ST instructions.

Branch instructions

opcode	Register	Offset
6 bits	5 bits	21 bits

Instruction	opcode	Register	Offset
BR	001100	R0	21 bits offset
BMI	001101	rs	21 bits offset
BPL	001110	rs	21 bits offset
BZ	001111	rs	21 bits offset

Stack Instructions

I-type encoding

Instructions	Opcode (6)	R1 (5)	R2 (5)	Immediate value(16)
PUSH	010000	SP	rs	XXXXXXX
POP	010001	SP	SP	00…0100
CALL	010010	xxxxx	xxxxx	Offset value
RET	010011	xxxxx	xxxxx	Xxxxxxxx

- When performing a POP operation, the ALU will compute SP = SP-4 and write the results to the register file.
- Write the data in R2 in the new SP for the push operation, where SP = SP+4.

Register to Register transfer instructions:

Use R-type instructions encoding

Instructions	opcode	R1	R2	R3	Extended opcode
MOVE	010100	rs	rt	xxxxx	xxxxxxxxxxxxxx

Program Control Instructions

Instruction	opcode
HALT	010101
NOP	010110

Overall table for opcode vs instructions

Opcode	Instruction
000000	ADD
000000	SUB
000000	AND
000000	OR
000000	XOR
000000	NOT
000000	SLA
000000	SRA
000000	SRL
000001	ADDI
000010	SUBI
000011	ANDI
000100	ORI
000101	XORI
000111	SLAI
001000	SRAI
001001	SRLI
001010	LD/LDSP
001011	ST/STSP
001100	BR
001101	BMI
001110	BPL
001111	BZ
010000	PUSH
010001	POP
010010	CALL
010011	RET
010100	MOVE

010101	HALT
010110	NOP

Branch Control Truth Table

Opcode	Sign	Carry	Zero	Conditional Flag
001100	X	X	X	1
001111	0	0	1	1
001111	0	1	X	0
001111	1	X	X	0
001101	1	1	X	1
001101	0	0	X	0
001101	0	1	X	0
001101	1	0	X	0
001010	0	0	0	1
001010	0	1	X	0
001010	1	0	X	0
001010	1	1	X	0
Default	X	X	X	0

- Opcode is the specific opcode value being tested, corresponding to the input opcode to the branch control module.

- Sign , carry , zero are the input for the sign , carry and zero signals , respectively, provided to the branch control module.
- Conditional flag is the output value of the conditional flag signal from the branch control module, indicating whether a jump is valid(1) or not (0) based on the conditions in the module.