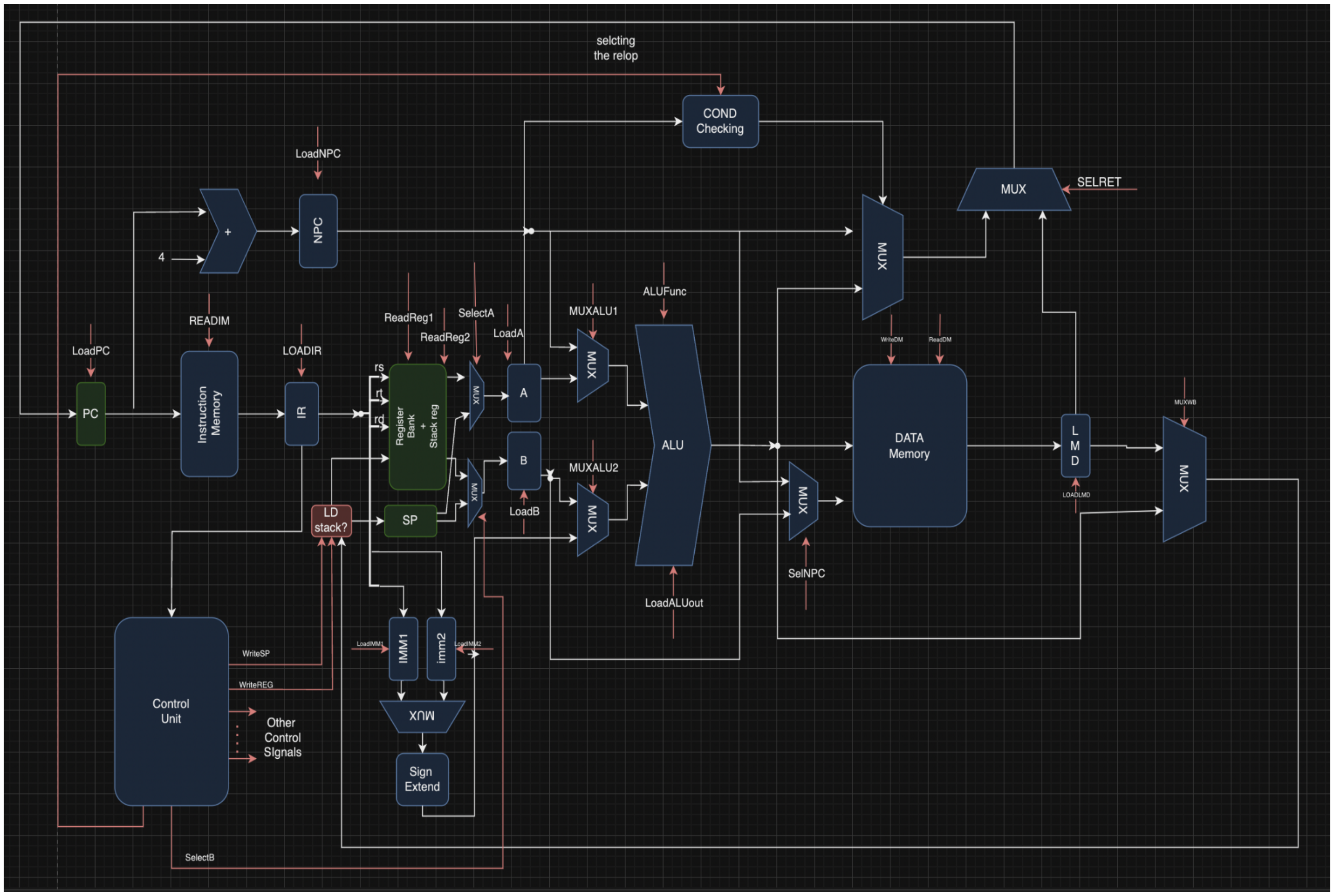
32 Bit Processor Design

Group 80

Navaneeth Shaji (21CS30032)
Niranjana Reddy B S (21CS30033)

Data Path and Control Signals



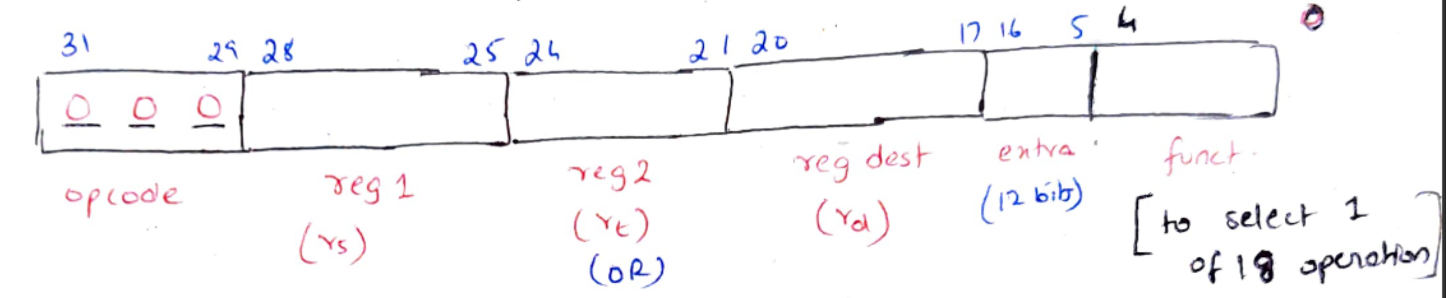
Instruction Encoding

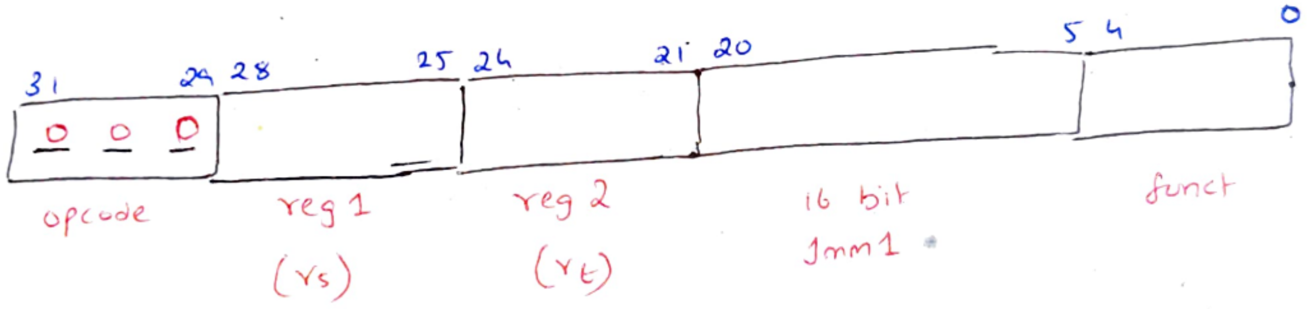
32 bits

4 bits for registers

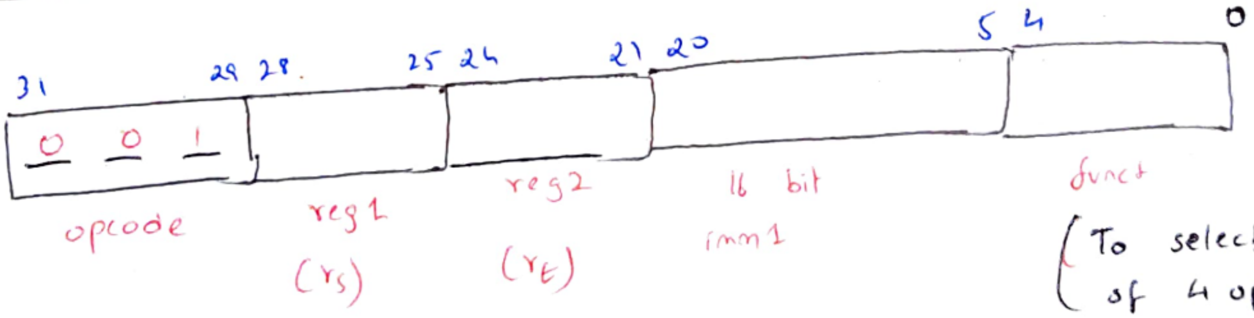
1 PC , 1 SP

a) ALU operations





b) Load / Store



(To select 1 of 4 operation)

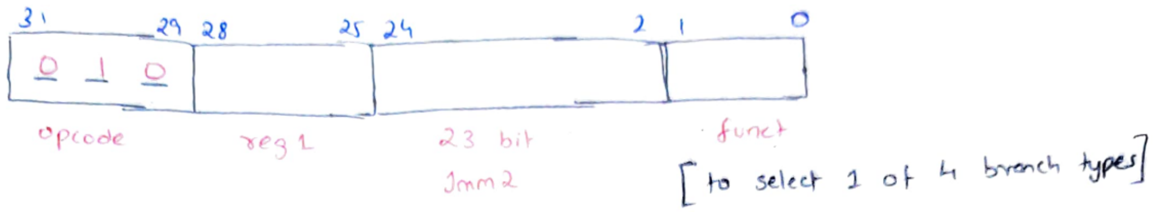
2 bits for them

1 bit to ~~the~~ do

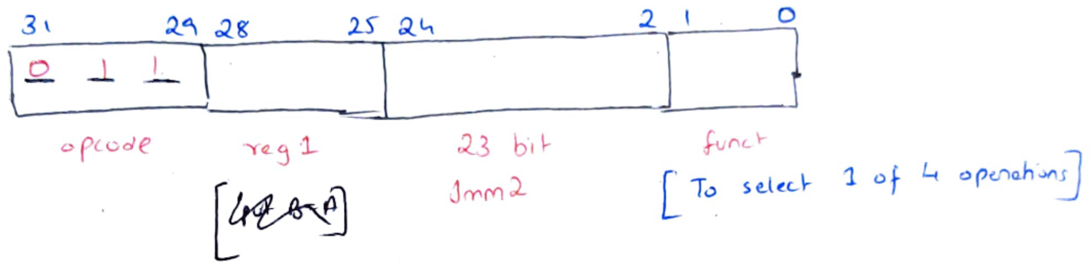
the select A

to decide A = SP or rs

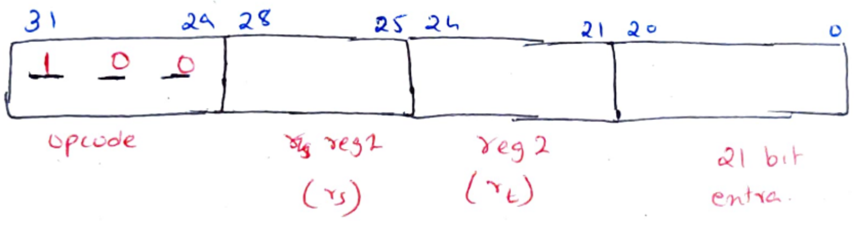
c) Branch operations



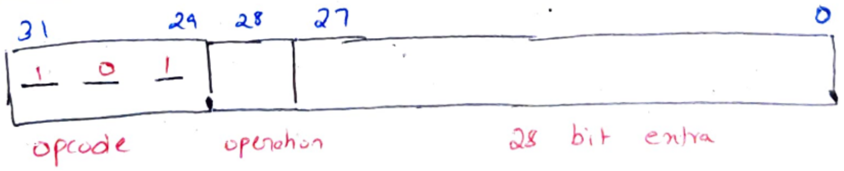
d) Stack operations



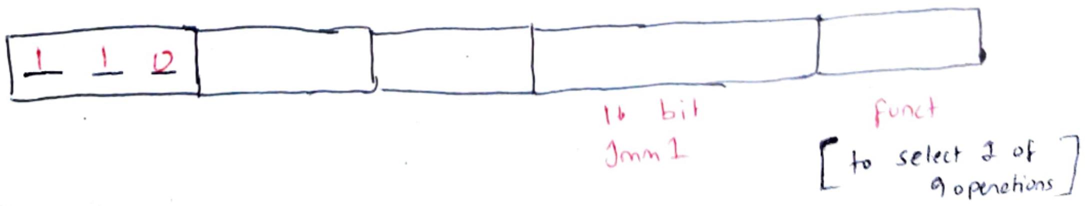
e] Move



f) Program Control



g) Stack ALU



Components Of the Design

Registers :

Sixteen 32-bit Registers(can be individually identified by a 4-bit code)

Stack Pointer Register(SP)

Program Counter(PC)

(NPC)

Instruction Register(IR)

Register A(Holds the first parameter send into ALU)

Register B(Holds the second Parameter send into ALU)

RegsiterIMM1(Holds the 16-bit Immediate Data)

RegisterIMM2(Holds the 23-bit immediate data)

Register LMD(Stores the value loaded from memory)

MUX:

MUX_A(Select the Parameter Loaded into Register A)

MUX_B(Select the parameter Loaded into Register B)

MUX_IMM(Selected the Immediate Data to be sign Extended)

MUX_ALU1(Selects the 1st Operand of ALU)

MUX_ALU2(Selects the 2nd operand of the AlU)

MUX_NPC(Selects between the NPC and B)

MUX_PC_RET(Selects the values the be loaded into PC)

MUX_WB(Selects the Write Back value)

MUX_BR(Selects if Npc or Alu output is selected during branch instructions)

Other Components:

ALU(Arithmetic LogicalUnit)

PC_INC(Increments the value of PC)

COND_CHECK(Used in Branching while Condition check)

CONTROL_UNIT(Relays the respective Control signals)

SIGN_EXTEND(Used to extend the immediate data)

Control Signals

These are all the control signals that are being used :

1.*LoadPC*
2. *LoadNPC*
3. *READIM*
4. *LOADIR*
5. *WriteREG* .
6. *WriteSP*
7.*ReadReg1*
8.*ReadReg2*
9.*LoadIMM1* .
10. *LoadIMM2*
11. *SelectA*
12. *SelectB*
13. *LoadA*
14. *LoadB*
1. *MUXALU1*
16. *MUXALU2*
17. *ALUFunc*
18. *LOADALUout*
19. *SelRELOP*
20. *WriteDM*
21. *ReadDM*
22. *LoadLMD*
23. *MUXWB*
24. *SelNPC*
25. *SelRET*

Operations and their corresponding Opcodes and Functional Opcodes

ALU operations

Operation	OPCODE (in decimal)	FUNCOP (in decimal)
ADD	0	0
SUB	0	1
AND	0	2
OR	0	3
XOR	0	4
NOT	0	5
SLA	0	6
SRA	0	7
SRL	0	8
ADDI	0	16
SUBI	0	17
ANDI	0	18
ORI	0	19
XORI	0	20
NOTI	0	21
SLAI	0	22
SRAI	0	23
SRLI	0	24

Load-Store Instructions

Operation	OPCODE (in decimal)	FUNCOP (in decimal)
LD	1	0
ST	1	1
LDSP	1	2
STSP	1	3

Branch Instructions

Operation	OPCODE (in decimal)	FUNCOP (in decimal)
BR	2	0
BMI	2	1
BPL	2	2
BZ	2	3

Move Instructions

Operation	OPCODE (in decimal)	FUNCOP (in decimal)
MOVE	4	NA

Program Control Instructions

Operation	OPCODE (in decimal)	PC_opcode
HALT	5	0
NOP	5	1

Stack ALU instructions

Operation	OPCODE (in decimal)	FUNCOP (in decimal)
ADD	6	0
SUB	6	1
AND	6	2
OR	6	3
XOR	6	4
NOT	6	5
SLA	6	6
SRA	6	7
SRL	6	8

Sample Instructions Execution with Control Signals

In the google Sheet below , we have shown the control signals that get activated for each of the time steps for some of the instruction .
We have chosen the instruction such that control signals for all the type of operations are shown at least once. Some of the instruction can be taken to be pseudo instruction and can be spilt into simpler instructions ,

For eg :

1. Push R6 can be done as SUBI SP,4 and ST R6,0(sp)
2. Call 1000 can be done as SUBI SP,4 , ST NPC,0(SP) and BR 1000

Multiple Instruction encodings will be generated to handle these kind of operations .

The control signals generated for each kind of operation in each of the 5 stages are tabulated in the following sheet : Control Unit