University of Moratuwa
Faculty of Engineering
Department of Electronic & Telecommunication Engineering

EN2031 - Fundamentals of Computer Organization and Design
Processor Design Project

Team Invictus

Index No	Name
200119C	D.K.D.Dewagiri
200222K	S.H.D.Himeka
200287L	P.B.K.S.G.Kapukotuwa

1) **Question (a) part (i) - The required instructions**

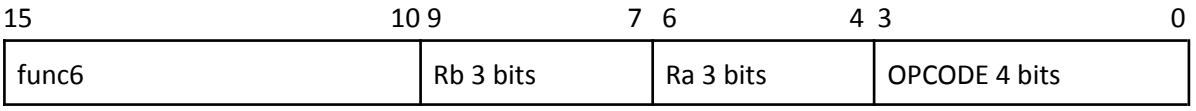
Type of Instruction	**Opcode**	**Process**	**Meaning**
ALU Operations	ADD Ra Rb	Ra←Ra+Rb	Add Rb and Ra and save it to the Ra
	SUB Ra Rb	Ra←Ra-Rb	Subtract Rb from Ra and save it to Ra
	AND Ra Rb	Ra ← Ra & Rb	Perform BITWISE AND operation on Ra and Rb and save it to Ra
	OR Ra Rb	Ra←Ra||Rb	Perform BITWISE OR between Ra and Rb and save it to Ra
Stack Operations	PUSH Ra	R7← R7+1 DM[R7] ←Ra	Push Ra to the top of stack
	POP Ra	Ra ← DM[R7] R7 ← R7-1	Pop top of the stack and store it in Ra
Load Operations	LDI Ra Rb	Ra ← DM[Rb]	Load data from memory address Rb and store in Ra
	LDIPM Ra Rb	Ra ← DM[Rb] Rb← Rb+1	Load data from memory address Rb and store in Ra and update the memory
	LUI Ra Imm8	Ra ←Ra+Imm8	Add immediate value to Ra and store in Ra
Store Operations	ST Ra Rb	DM[Ra] ← Rb	Store the data in Rb to the memory address in Ra
	STPM Ra Rb	DM[Ra] ← Rb Ra ←Ra+1	Store the data in Rb to the memory address in Ra and update the memory
Control Operations	JMP Imm8	PC ← PC+Imm8	Jump (unconditional branch)
	BEQ Ra R6 I		Branch if equal
	BLTEQ Ra I		Branch if less than or equal
	BGTEQ Ra I		Branch if greater than or equal
Data Transfer	MOVE Ra Rb	Ra ← Rb	Move from Rb to Ra

2) Question (a) part (ii) - Instruction formats required

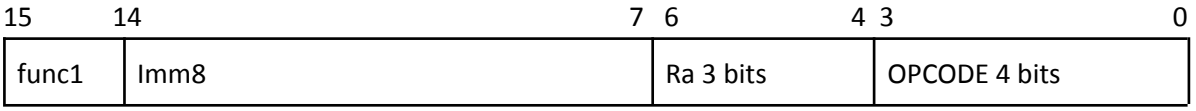
Total instructions 16 → 4 bits
Registers 8 → 3 bits
Immediates →8 bits
Data memory → 12 bits

Also note that Ra is the destination register and Rb is the source register. They can be any register from R0 to R5.

i) ALU Operations

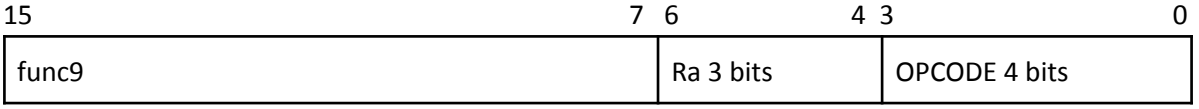


ii) Control Flow

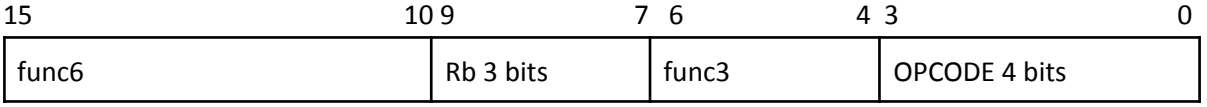


iii) Stack Operations

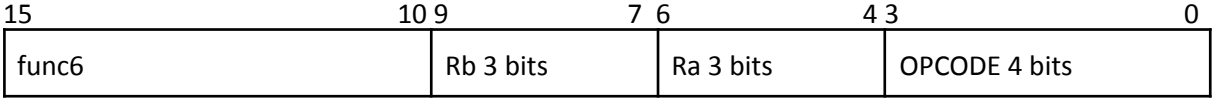
POP



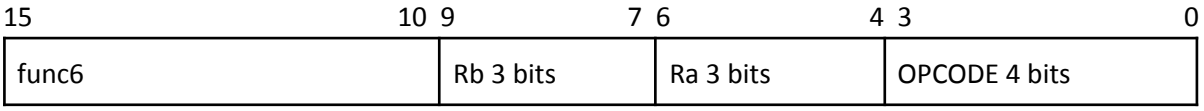
PUSH



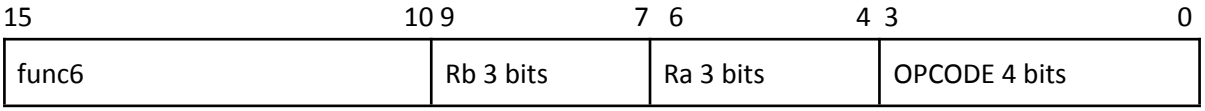
iv) Load Operations



v) Store Operations



vi) Data Transfer



When deciding the instruction format, the number of bits reserved for the sections opcode, Ra, Rb, Imm8 depended on the number of registers and opcodes available.
In ALU, PUSH, Load, Store and Move instructions, The first 4 bits were fixed and assigned for the opcode for the ease of decoding. The next 3 bits are reserved for Ra, next 3 for Rb and the remaining 6 bits are kept as function 6, to be used if needed in the future.
In control flow instructions the first 4 bits are reserved for opcode, the next 3 for the register, the next 8 for immediate value. The remaining 1 bit is left as function1.
In the POP instruction format, the first 4 bits are used for the opcode and the next 3 are used for the source register.

3) Question (b) - OPCODE encoding scheme for easy decoding

inst[2:1] inst[4:3]	00	01	11	10
00	ADD	SUB	AND	OR
01	PUSH	POP	LDI	LDIPM
11	BEQ	BLTEQ	BGTEQ	JMP
10	LUI	ST	MOV	STPM

ALU OPCODES — starts with 00
0000 → ADD
0001 → SUB
0010 → OR
0011 → AND

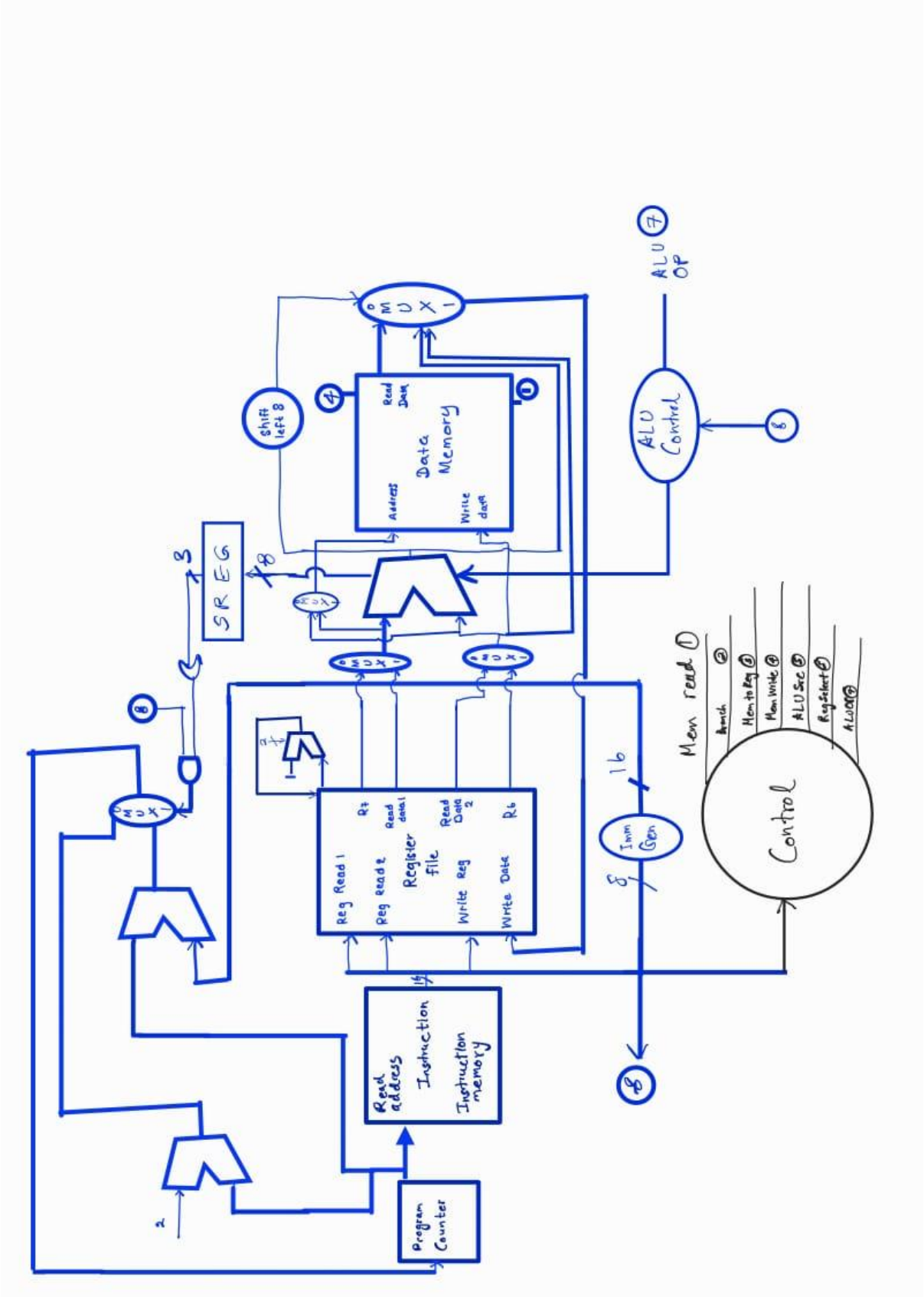
STACK OPERATION OPCODES — starts with 01
0100 →PUSH
0101→POP

LOAD AND STORE OPCODES
0110 →LDIPM
0111 →LDI
1000 →LUI
1001 →ST
1010 →STPM

DATA TRANSFER OPCODES
1011 → MOV

CONTROL FLOW OPCODES — starts with 11
1100 → BEQ
1101 → BLTEQ
1110 →BGTEQ
1111 → JMP

4) Question (c) - Datapath and the control signals



5) Question (d) part (i) - The chosen design approach for the microarchitecture

We've decided to choose hardwired micro architecture for this processor. The reasons why we selected the particular architecture are as follows.

- Simple Design

  This architecture has a fixed set of instructions. It does not need a control memory to generate control signals. So the complexity of this approach is far lower when compared with the micro programming architecture.

- Speed Action

  Control signals are directly generated from the Opcde. No need for microprogramming.

- Smaller size

  This approach requires less hardware than microprogramming. It is beneficial having considering the cost concerns.

- Better power efficiency

  Hardwired architecture needs less overheads in terms of decoding and execution.

- Can be customized

  More customisation is possible with hardwired designs than with microprogrammed systems. This is so that each instruction may be tailored to the particular needs of the system as it is directly implemented in hardware.

6) Question (d) part (ii) - Controller for the datapath

		Branch	MemRead	MemtoReg	ALUOp	MemWrite	ALUSrc*	RegWrite	RegSelect	RegSelect2 01
ADD		0	0	0	1	0	0	1	0	0
SUB		0	0	0	1	0	0	1	0	0
AND		0	0	0	1	0	0	1	0	0
OR		0	0	0	1	0	0	1	0	0
POP		0	1	1	1	0	0	1	1	0
PUSH		0	0	0	1	1	0	0	1	0
MOVE		0	0	0	0	0	1	1	1	1
LDI		0	1	1	0	0	0	1	0	0
LDIPM		0	1	1	1	0	1	1	0	0
ST		0	0	0	0	1	1	0	0	0
STPM		0	0	0	1	1	1	0	0	0
LUI		0	0	0	1	0	1	1	1	0
JMP		1	0	0	1	0	0	0	0	0
BEQ		1	0	0	1	0	0	0	1	1
BLTEQ		1	0	0	1	0	0	0	1	1
BGTEQ		1	0	0	1	0	0	0	1	1