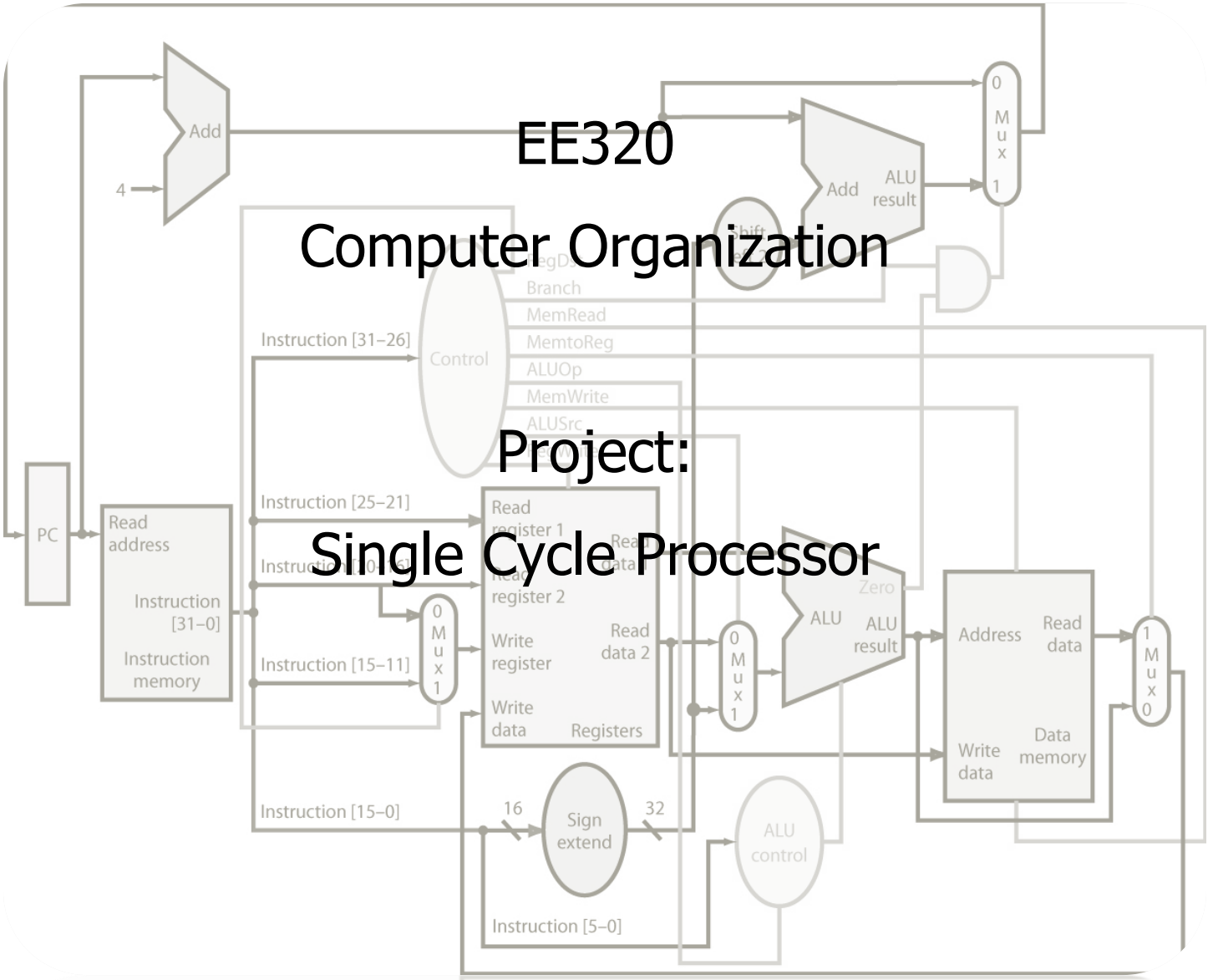
EE320

Computer Organization



Project:

Single Cycle Processor

Kumail Haider (13100183)

Amash Malik (13100195)

Taha Khan (13100158)

Bilal Shahid (13100148)

Table of Contents

1 Objectives Achieved:

- To develop a model of a single cycle processor following the principles as laid out in MIPS computer architecture

- To design various instructions given the limitations for an 8-bit processor and an 8-bit register file

- To outline how the hardware implements the computer instructions and the way they are processed through various parts such as instruction memory, data memory and register file, i.e., understanding the data path that they follow.

2 Project Specifications

Size of instructions: 16 bits

Number of Instructions:

4 R type: add, sub, and, or (**nor implemented as a bonus**)

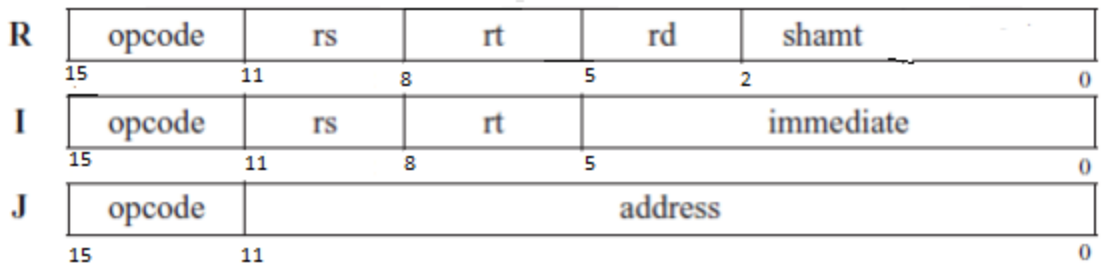
3 I type: addi, andi, ori

2 Data transfer instructions: lw,sw

2 Branch Instructions: beq, bne

1 J type: j

2.1 Instruction formats:



The R type format was implemented as a 16-bit instruction with 3 bits each for Rs, Rt, Rd registers. The funct field was deleted since it was incorporated within the opcode field which determined the instruction to execute. For the I type the immediate field was 6 bits long. The j type was implemented by a 12 bit address field and a 4 bit opcode.

2.2 Design Overview of 8-bit MIPS Processor:

The instruction format is described as before. Two 8-bit ROMS serve as the instruction memory. The upper 4 bits, i.e. the opcode are sent to control. The 3 bit addresses of registers are fed to the 8 bit Register file. The register file sends the 8 bit data for ALU operations and respective instructions as appropriate. The 6 bit immediate will be sent to Sign extension. The Muxes select appropriate values. Mux 1 selects between Rt and Rd. Mux 2 selects between immediate and Rt value. Mux 3 selects whether to read/write data from memory or write back to register file.

Branch Control Logic

Input	X0	X1	Decision
0	0	X	No Branch
0	0	X	No Branch
Z	1	0	BEQ
~Z	1	1	BNE

The table above shows the decision taken at branching MUX. In general, when X0 will be 0, the decision will always be "No Branching". So X1 will be "don't care" when X0 is 0. X1 chooses which branch condition to take, ie either beq or bne as shown in the table above. The output is the branch decision which is connected to MUX 4. When branch is taken MUX 4 selects one to which the immediate is connected.

3 Control Truth Table

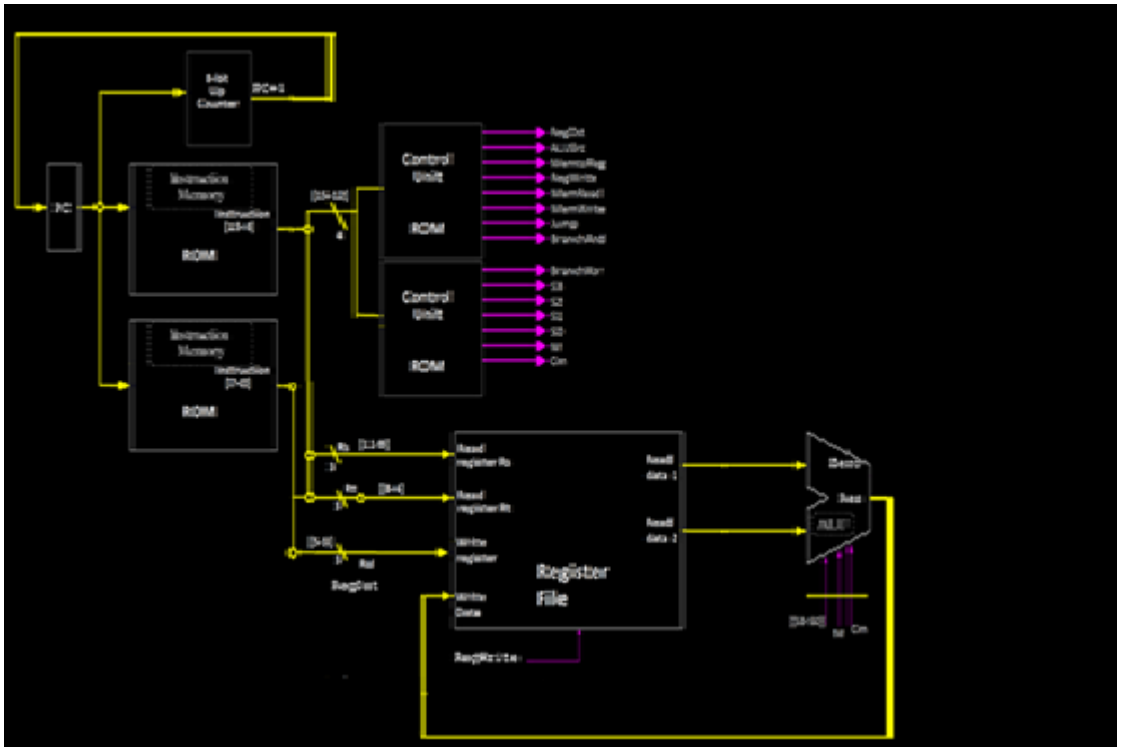
Truth Table for the control of the complete instruction set:

Instruction	Mux1	Mux2	Mux3	Mux5	To branch control logic		Register Write	Mem R/W	Opcodes	ALUOP						Sign Extension
					X0	X1				M	CN	S3	S2	S1	S0	
add	0	0	0	0	0	0	1	0	0000	0	1	1	0	0	1	0
sub	0	0	0	0	0	0	1	0	0001	0	0	0	1	1	0	0
or	0	0	0	0	0	0	1	0	0010	1	0	0	1	1	1	0
and	0	0	0	0	0	0	1	0	0011	1	0	1	1	0	1	0
nor	0	0	0	0	0	0	1	0	0100	D	D	D	D	D	D	0
addi	1	1	0	0	0	0	1	0	0101	0	1	1	0	0	1	1
andi	1	1	0	0	0	0	1	0	0110	1	0	1	1	0	1	0
ori	1	1	0	0	0	0	1	0	0111	1	0	0	1	1	1	0
lw	1	1	1	0	0	0	1	0	1000	0	1	1	0	0	1	1
sw	1	1	0	0	0	0	0	1	1001	0	1	1	0	0	1	1
bne	1	1	0	0	1	1	0	0	1010	0	0	0	1	1	0	1
beq	1	1	0	0	1	0	0	0	1011	0	0	0	1	1	0	1
slt	0	0	0	0	0	0	1	0	1100	D	D	D	D	D	D	0
jump	X	1	0	1	0	0	0	0	1101	X	X	X	X	X	X	0

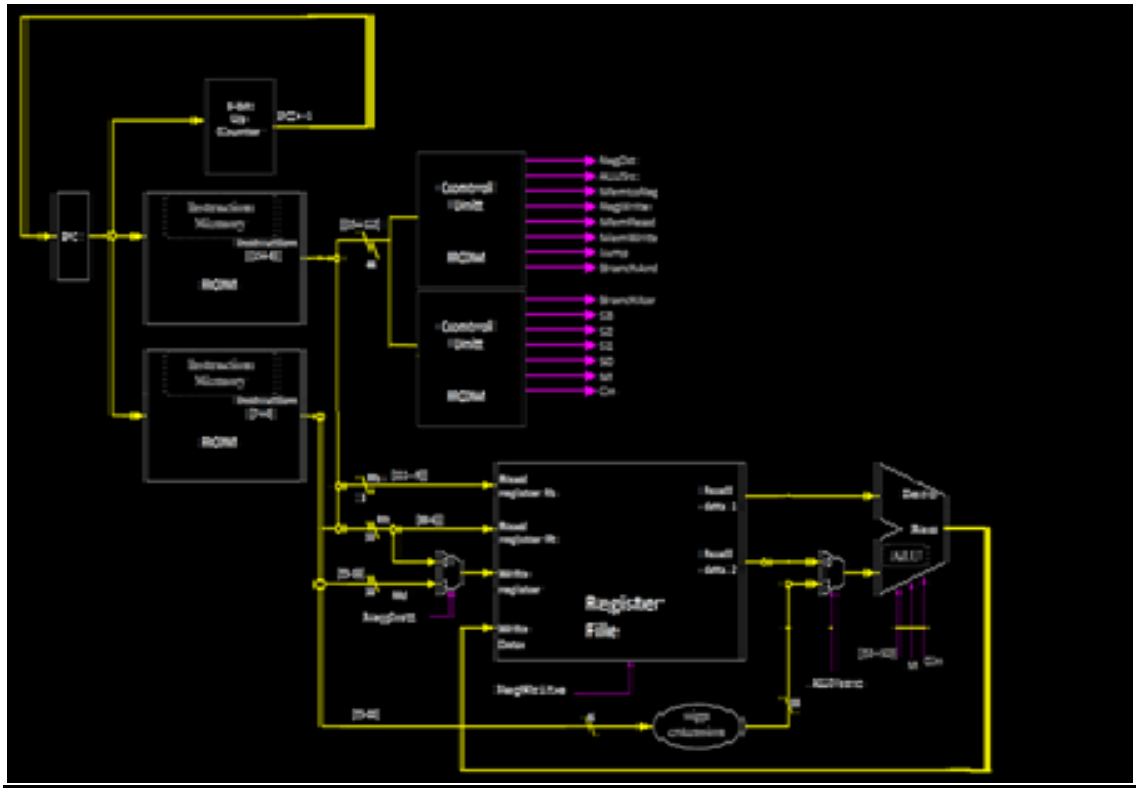
X = don’t care

4 Block Diagram of the Complete Processor

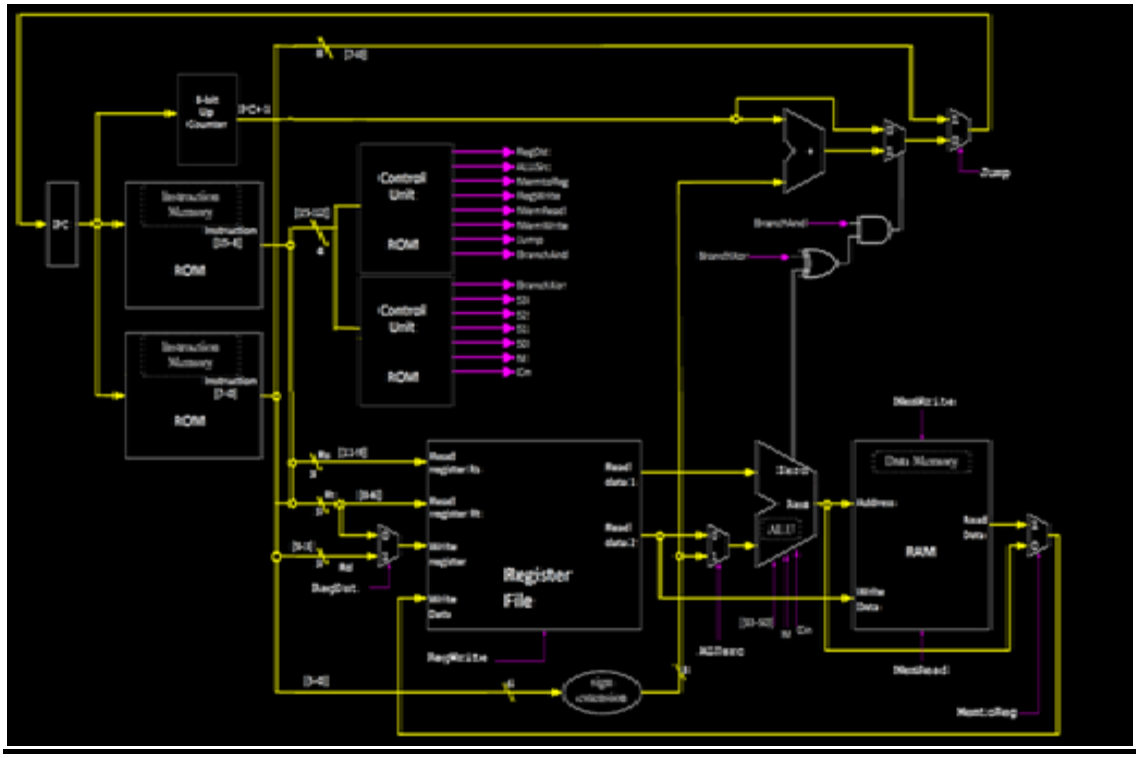
4.1 R type



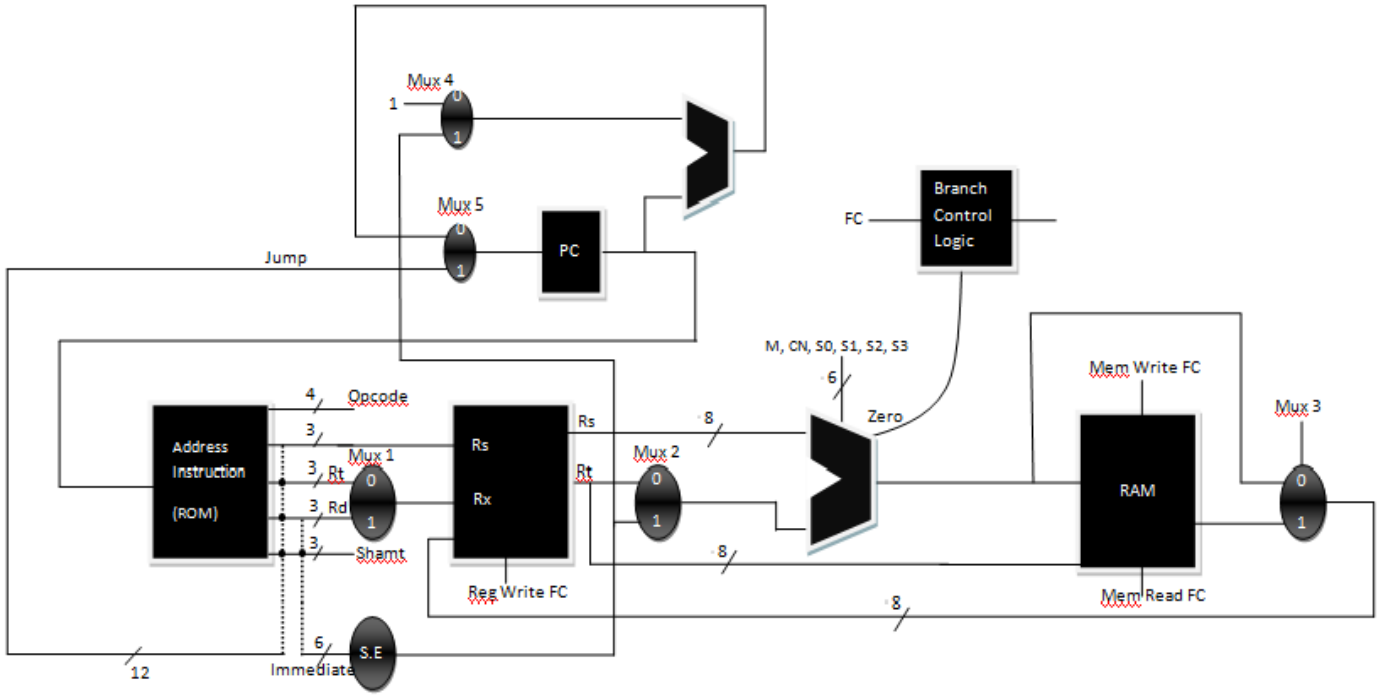
4.2 I Type



4.3 J Type (incorporating BNE/BEQ and LW/SW)



4.4 Complete Processor (minute details not shown)



FC= from control

5 Innovation (Things achieved in a different manner)

Program Counter:

The PC was implemented by an 8 bit register. Current address was incremented by connecting PC output to an ALU in A+B (addition) mode. The other input of this ALU was connected to the MUX 4, whose select line was coming from the branch control logic. When there was no branch, the MUX selected constant 1. Hence we get PC=PC+1.

When branch condition was satisfied, the control of MUX 4 selected input coming from the sign extended immediate. So the PC = PC + branch offset was given.

We implemented branching with respect to current address and not current address + 1.

6 Problems and Issues, and how we resolved

1. The biggest issue was: ROMs. Sometimes the chips didn't work and sometimes the programmer didn't work as it should. This was because we were dependent on programming these chips again and again.We resolved this issue by reducing our reliance on programming the chips again and again. We formed our Complete Truth Table of the project control at the time of proposal submission. So we did NOT program control ROMs for each module submission. We programmed the complete control of our project into our Control ROMs at one time.

2. We made a clean circuit: unorganized wiring initially lead to transients that caused strange behaviour of various chips. But, using PCBs that optimized the pace also saved us from extra and too many buses unlike other groups. For example, we used MUXes on the same PCB that had ALU control etc. to avoid too many buses jutting out.

However, there was no major issue with regards to the design of the processor.

7 Further Enhancements

This SCP can be modified further to achieve custom-designed instructions, for example, a new type of instruction that uses three registers and adds their content to a fourth register like:

Addt $a0, $a1, $a2, $a3

With a slight modification to the ALU and MUXes, a new instruction format may be achieved.

Again, the processor designed in this project forms the basis for the 5 stage integer pipelined MIPS processor and hence it further provides information on how dual-core processors and other microprocessors work.