Harvard Architecture based 8-bit Processor Design

1 Problem Statement

Design an 8-bit processor (single core) with the following specifications.

- 32-numbers of 8-bit registers.
- 8-bit logic and unsigned fixed point arithmetic circuit.
- Load and Store unit.
- Data memory with 256 numbers of 8-bit locations.
- 32-bit Instructions.
- Instruction memory with 64 numbers of 32-bit locations.
- 6-bit Program Counter.

2 Design Approach

The ALU can be designed as follows.

2.1 Fixed Point Adder/Subtractor

Design 8-bit Carry Lookahead Adder circuit using KGP blocks and recursive doubling algorithm. Now, use the same module for the design of subtractor with 2's complement.

2.2 Fixed Point Multiplier

Design the 8-bit Wallace Tree multiplier or Carry Save Array multiplier using carry-save-adders.

2.3 Logical Left/Right Shifter

Design 8-bit Barrel or log logical left/right shifter.

2.4 Logic Unit

Design 8-bit Logic Unit to perform NOT, AND, OR, NAND, NOR, XOR, and XNOR operations.

3 Instruction Format and Encoding

3.1 Instruction Format

The design instruction should of 32-bit size. The immediate and register move instructions will have the format as shown in Figure 1(a) and (b) respectively. Similarly, the Load and Store instructions will have the format as shown in Figure 1(c) and (d) respectively. The rest of the logic and arithmetic instructions will have the format as shown in Figure 1(e).

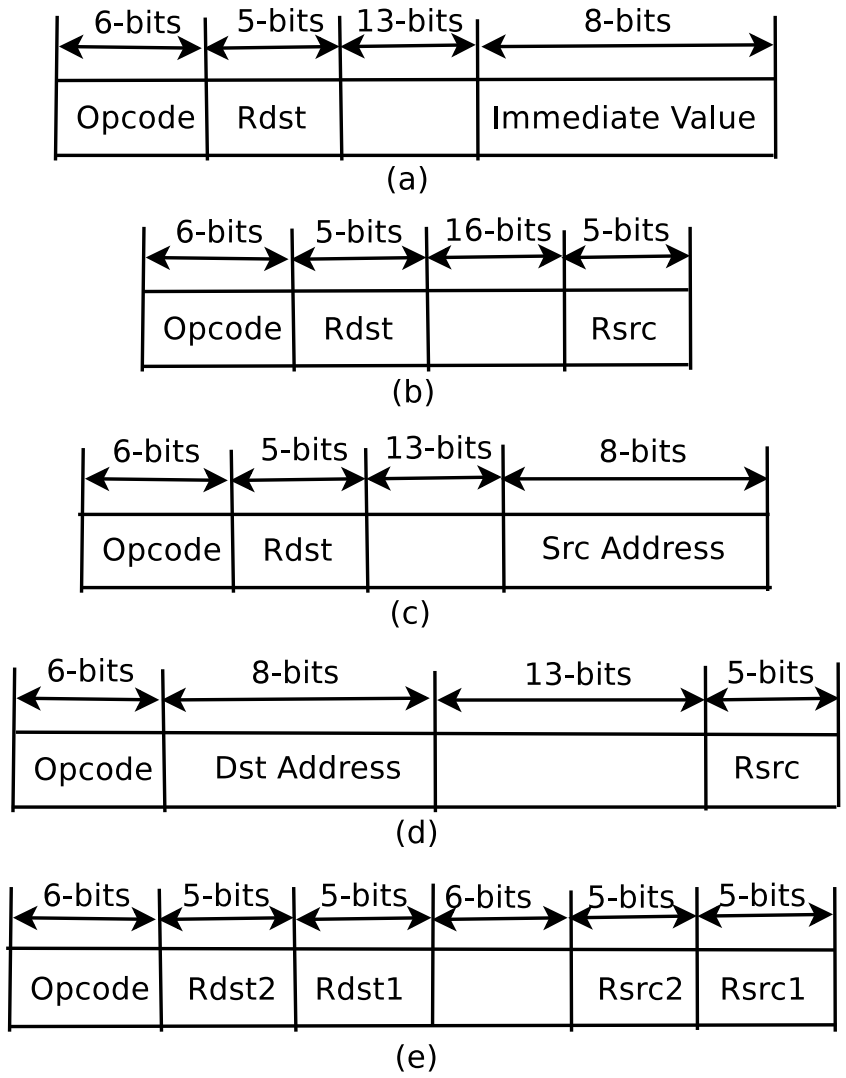

Figure 1: Instruction format for (a) Immediate move instructions, (b) register move instructions, (c) load instructions, (d) store instructions, and (e) arithmetic/logic instructions.

3.2 Operation code Encoding

Encode the instruction opcode as shown in Table 1.

4 Processor Architecture

The required processor architecture is shown in Figure 2, where the ALU includes all the logic and arithmetic functions units as mentioned in Section 1.

Table 1: Instruction details with opcode

Opcode	Description	Usage	Operation
000000	Immediate move	$MOV\ Rdst,\ \#Imm\_value$	$Rdst = Imm\_value$
000001	Register move	$MOV\ Rdst,\ Rsrc$	$Rdst = Rsrc$
000010	Load	$LOAD\ Rdst,\ [Src\ address]$	$Rdst = @[Src\ address]$
000011	Store	$STORE\ [Dst\ address],\ Rsrc$	$@[Dst\ address] = Rsrc$
000100	16-bit fixed point addition	$ADD\ Rdst1,\ Rsrc2,\ Rsrc1$	$Rdst1 = Rsrc2 + Rsrc1$
000101	16-bit fixed point subtraction	$SUB\ Rdst1,\ Rsrc2,\ Rsrc1$	$Rdst1 = Rsrc2 - Rsrc1$
000110	16-bit fixed point negation	$NEG\ Rdst1,\ Rsrc1$	$Rdst1 = -Rsrc1$
000111	16-bit fixed point multiplication	$MUL\ Rdst2,\ Rdst1,\ Rsrc2,\ Rsrc1$	$\{Rdst2, Rdst1\} = Rsrc2 \times Rsrc1$
001000	16-bit fixed point division	$DIV\ Rdst1,\ Rsrc2,\ Rsrc1$	$Rdst1 = Rsrc2/Rsrc1$
001001	16-bit Logical OR	$OR\ Rdst1,\ Rsrc2,\ Rsrc1$	$Rdst1 = Rsrc2 \mid Rsrc1$
001010	16-bit Logical XOR	$XOR\ Rdst1,\ Rsrc2,\ Rsrc1$	$Rdst1 = Rsrc2 \oplus Rsrc1$
001011	16-bit Logical NAND	$NAND\ Rdst1,\ Rsrc2,\ Rsrc1$	$Rdst1 = Rsrc2\ !\&\ Rsrc1$
001100	16-bit Logical NOR	$NOR\ Rdst1,\ Rsrc2,\ Rsrc1$	$Rdst1 = Rsrc2\ !\mid\ Rsrc1$
001101	16-bit Logical NXOR	$XNOR\ Rdst1,\ Rsrc2,\ Rsrc1$	$Rdst1 = Rsrc2\ !\oplus\ Rsrc1$
001110	16-bit Logical NOT	$NOT\ Rdst1,\ Rsrc1$	$Rdst1 = !Rsrc1$
001111	16-bit Logical left shift	$LLSH\ Rdst1,\ Rsrc2,\ Rsrc1$	$Rdst1 = Rsrc2 << Rsrc1$
010000	16-bit Logical right shift	$LRSH\ Rdst1,\ Rsrc2,\ Rsrc1$	$Rdst1 = Rsrc2 >> Rsrc1$

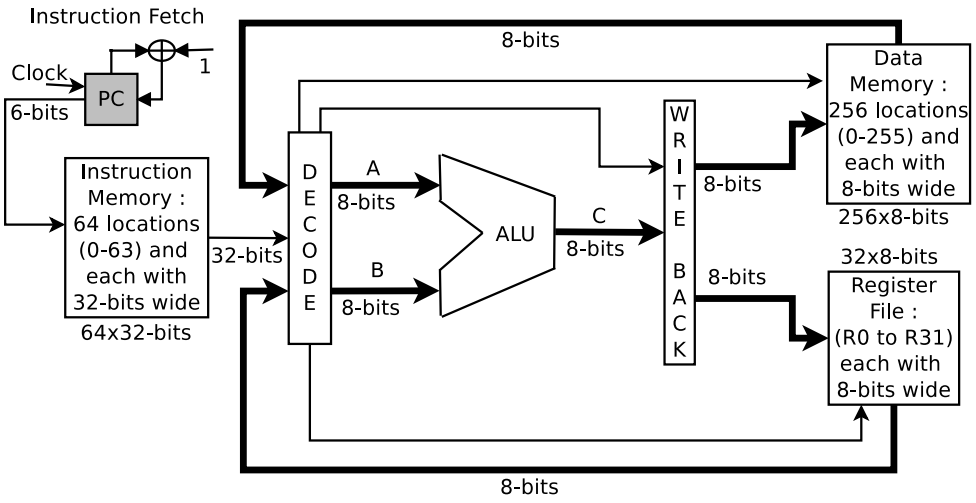

Figure 2: An 8-bit Harvard Processor Architecture