For **J-type** instructions, an 11-bit immediate constant is used for LUI (load upper immediate), J (jump), and JAL (jump-and-link) instructions.

Instruction Encoding

Nine R-type instructions, twelve I-type instructions, and three J-type instructions are defined. These instructions, their meaning, and their encoding are shown below:

Instr.	Meaning	Encoding				
ADD	Reg(Rd) = Reg(Rs) + Reg(Rt)	Op = 00000	f = 00	Rd	Rs	Rt
SUB	Reg(Rd) = Reg(Rs) – Reg(Rt)	Op = 00000	f = 01	Rd	Rs	Rt
SLT	Reg(Rd) = Reg(Rs) signed < Reg(Rt)	Op = 00000	f = 10	Rd	Rs	Rt
SLTU	Reg(Rd) = Reg(Rs) unsigned < Reg(Rt)	Op = 00000	f = 11	Rd	Rs	Rt
AND	Reg(Rd) = Reg(Rs) & Reg(Rt)	Op = 00001	f = 00	Rd	Rs	Rt
OR	Reg(Rd) = Reg(Rs) | Reg(Rt)	Op = 00001	f = 01	Rd	Rs	Rt
XOR	Reg(Rd) = Reg(Rs) ^ Reg(Rt)	Op = 00001	f = 10	Rd	Rs	Rt
NOR	Reg(Rd) = ~(Reg(Rs) | Reg(Rt))	Op = 00001	f = 11	Rd	Rs	Rt
SLL	Reg(Rd) = Reg(Rs) << Reg(Rt)	Op = 00010	f = 00	Rd	Rs	Rt
SRL	Reg(Rd) = Reg(Rs) zero>> Reg(Rt)	Op = 00010	f = 01	Rd	Rs	Rt
SRA	Reg(Rd) = Reg(Rs) sign>> Reg(Rt)	Op = 00010	f = 10	Rd	Rs	Rt
ROR	Reg(Rd) = Reg(Rs) rot>> Reg(Rt)	Op = 00010	f = 11	Rd	Rs	Rt
REVL	Reg(Rd[b7, …, b0]) = Reg(Rs[b0, …, b7])	Op = 00011	f = 00	Rd	Rs	000
REVH	Reg(Rd[b7, …, b0]) = Reg(Rs[8, …, b15])	Op = 00011	f = 01	Rd	Rs	000
REVA	Reg(Rd[b15, …, b0])= Reg(Rs[b0, …, b15])	Op = 00011	f = 10	Rd	Rs	000
JR	PC = Reg(Rs)	Op = 00011	f = 11	000	Rs	000
ADDI	Reg(Rt) = Reg(Rs) + Immediate^5	Op = 01000	Immediate^5		Rs	Rt
ANDI	Reg(Rt) = Reg(Rs) & Immediate^5	Op = 01001	Immediate^5		Rs	Rt
ORI	Reg(Rt) = Reg(Rs) | Immediate^5	Op = 01010	Immediate^5		Rs	Rt
XORI	Reg(Rt) = Reg(Rs) ^ Immediate^5	Op = 01011	Immediate^5		Rs	Rt
LW	Reg(Rt) = Mem(Reg(Rs) + Imm^5)	Op = 01100	Immediate^5		Rs	Rt
SW	Mem(Reg(Rs) + Imm^5) = Reg(Rt)	Op = 01101	Immediate^5		Rs	Rt
BEQ	Branch if (Reg(Rs) == Reg(Rt))	Op = 01110	Immediate^5		Rs	Rt
BNE	Branch if (Reg(Rs) != Reg(Rt))	Op = 01111	Immediate^5		Rs	Rt
J	**PC = (Zero-Extend)Immediate^{11}**	Op = 10000	Immediate^{11}			
JAL	R7 = PC + 1, **PC = (Zero-Extend)Immediate^{11}**	Op = 10001	Immediate^{11}			
LUI	R1 = Immediate^{11} << 5	Op = 11111	Immediate^{11}			

Opcodes 0, 1, 2, and 3 are used for R-type instructions. There are three shift and one rotate instruction. For the shift and rotate instructions, the lower 4 bits of register Rt are used as the shift/rotate amount. Opcode 3 is used for the JR (jump register) instruction and the bit-reversal instructions. Opcodes 8 through 15 are used for I-type instructions. The 5-bit immediate constant is zero-extended for ANDI, ORI, and XORI. It is sign-extended for the remaining instructions. The J-

type instructions have an 11-bit immediate constant. The Load Upper Immediate (LUI) is of the J-type to have an 11-bit immediate constant loaded into the upper 11 bits of register R1. The LUI can be combined with ORI to load any 16- bit constant into a register. Although the instruction set is reduced, it is still rich enough to write useful programs. Procedure calls and returns are implemented using the JAL and JR instructions.

Note about the Reverse Instructions:
The instructions REVL, REVH and REVA are used to reverse the contents of the source register (Rs) and save the result in the destination register (Rd). Details of those instructions are given in the table below.

Instruction	Syntax	Effect	Result
REVA	REVA Rd, Rs	Reverses the contents of Rs and saves the result in Rd.	Reg(Rd[b15, …, b0]) ← Reg(Rs[b0, …, b15])
REVL	REVL Rd, Rs	Reverses the contents of the lower byte of Rs and saves the result in the lower byte of Rd.	Reg(Rd[b7, …, b0]) ← Reg(Rs[b0, …, b7])
REVH	REVH Rd, Rs	Reverses the contents of the higher byte of Rs and saves the result in the lower byte of Rd.	Reg(Rd[b7, …, b0]) ← Reg(Rs[b8, …, b15])

Memory

The processor will have separate instruction and data memories with 2^{16} words each. Each word is 16 bits or 2 bytes and the Memory is *word addressable*. Only words (not bytes) can be read and written to memory, and each address is a word address. The purpose of this addressing is to simplify the implementation. The PC contains a word address (not a byte address); therefore, it is sufficient to increment the PC by 1 (rather than 2) to point to the next instruction in memory. Also, the Load and Store instructions can only load and store words. There is no instruction to load or store a byte in memory.

Register File

Implement a Register file containing Seven 16-bit registers R1 to R7 with two read ports and one write port. R0 is hardwired to zero.

Arithmetic and Logical Unit (ALU)

Implement a 16-bit ALU to perform all the required operations:
ADD, SUB, SLT, SLTU, OR, AND, XOR, NOR, SLL, SRL, SRA, ROR, REVA, REVL and REVH.

Addressing Modes

PC-relative addressing mode is used in the CPU for branch and jump instructions.
For branching (BEQ, BNE), the branch target address is computed as follows:
PC = PC + sign-extend(Imm5). Add the contents of PC to the sign-extended 5-bit Immediate. For jumps (J and JAL): PC = PC + sign-extend(Imm11).
For LW and SW base-displacement addressing mode is used. The base address is obtained from register (Rs) and added to the sign-extended 5-bit immediate to compute the memory address.