Intructions

A. Memory access
1. Cargar palabra:
   LD ws, offset(rs1)
   ws:=Mem16[rs1 + offset]
2. Guardar palabra:
   ST rs2, offset(rs1)
   Mem16[rs1 + offset]=rs2

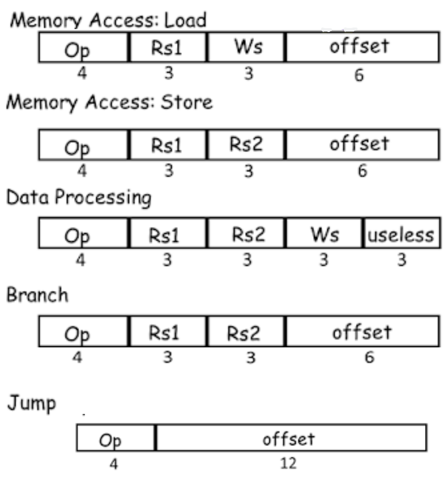
B. Data Processing Instructions

1. Add:
   ADD ws, rs1, rs2
   ws:=rs1 + rs2
2. Subtract:
   SUB ws, rs1, rs2
   ws:=rs1 – rs2
3. Invert (1's complement):
   INV ws, rs1
   ws:=!rs1
4. Logical Shift Left:
   LSL ws, rs1, rs2
   ws:=rs1 << rs2
5. Logical Shift Right:
   LSR ws, rs1, rs2
   ws:=rs1 >> rs2
6. Bitwise AND:
   AND ws, rs1, rs2
   ws:=rs1 • rs2
7. Bitwise OR:
   OR ws, rs1, rs2
   ws:=rs1 | rs2
8. Set on Less Than:
   SLT ws, rs1, rs2
   ws:=1 if rs1 < rs2; ws:=0 if rs1 ≥ rs2

C. Control Flow Instructions

1. Branch on Equal:
   BEQ rs1, rs2, offset
   Branch to (PC + 2 + (offset << 1)) when rs1 = rs2
2. Branch on Not Equal:
   BNE rs1, rs2, offset
   Branch to (PC + 2 + (offset << 1)) when rs1 != rs2
3. Jump:
   JMP offset Jump to {PC [15:13], (offset << 1)}

Format:



Opcode

0000	Load Word
0001	Store Word
0002	Add
0003	Subtract
0004	Invert (1's complement)
0005	Logical Shift Left
0006	Logical Shift Right
0007	Bitwise AND
0008	Bitwise OR
0009	Set on Less Than
0010	Hamming Distance
0011	Branch on Equal
0012	Branch on Not Equal
0013	Jump

ALU Control				
ALUOp	Opcode(hex)	ALUcnt	ALU Operation	Instruction
10	xxxx	000	ADD	LW,SW
01	xxxx	001	SUB	BEQ,BNE
00	0002	000	ADD	D-type: ADD
00	0003	001	SUB	D-type: SUB
00	0004	010	INVERT	D-type: INVERT
00	0005	011	LSL	D-type: LSL
00	0006	100	LSR	D-type: LSR
00	0007	101	AND	D-type: AND
00	0008	110	OR	D-type: OR
00	0009	111	SLT	D-type: SLT

Control signals									
Instruction	Reg Dst	ALU Src	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp	Jump
Data-processing	1	0	0	1	0	0	0	00	0
LW	0	1	1	1	1	0	0	10	0
SW	0	1	0	0	0	1	0	10	0
BEQ,BNE	0	0	0	0	0	0	1	01	0
J	0	0	0	0	0	0	0	00	1