

COA Mini Project

Encoding Scheme

Submitted by : Om Saran (20CS02007)
 Yatin Dhiman (20CS02010)

32 Bit Instruction : XXXX DDDD AAAA BBBB Iiiiiiiiiiiiiiii

Instruction Format Specification :

Component	Binary	Hexadecimal
OPCODE	XXXX	X
Destination Register (Rd)	DDDD	D
Source Register 1 (R1)	AAAA	A
Source Register 2(R2)	BBBB	B
Immediate Value	Iiiiiiiiiiiiiiii	IIII

In this design, we have used 8 32-bit Registers, so we need 4 bits to identify each register uniquely, as represented in the following table:

Registers Encoding:

Register	Binary Coding	Hexadecimal Encoding
R0	0000	0
R1	0001	1
R2	0010	2
R3	0011	3
R4	0100	4
R5	0101	5
R6	0110	6
R7	0111	7

All the instructions supported by our 32-bit RISC processor are identified by a unique code known as OPCODE, which are the first 4 bits of a 32-bit instruction. As per problem statement various instructions supported by the design are :

1. MOVE R_i, R_j \\The content of R_j is transferred to R_i .
2. MOVE R_i , Immediate (16-bit) \\The immediate value (32-bit unsigned extended) will be transferred to R_i .
3. LOAD $R_i, X(R_j)$ \\ The content of memory location $[[R_j] + X]$ is loaded into R_i , where X is a 16-bit unsigned immediate value.

4.STORE $R_i, X(R_j)$ $\backslash\backslash$ The content of register R_i is stored in memory $[[R_j] + X]$, where X is a 16-bit unsigned immediate value.

5.ADD R_i, R_j, R_k $\backslash\backslash R_i = R_j + R_k$.

6.ADI $R_i, R_j, \text{Immediate (16-bit)}$ $\backslash\backslash R_i = R_j + \text{Immediate Value}$
(32-bit unsigned extended)

7.SUB R_i, R_j, R_k $\backslash\backslash R_i = R_j - R_k$

8.SUI $R_i, R_j, \text{Immediate (16-bit)}$ $\backslash\backslash R_i = R_j - \text{Immediate Value}$
(32-bit unsigned extended)

9.AND R_i, R_j, R_k $\backslash\backslash R_i = R_j \text{ AND } R_k$.

10.ANI $R_i, R_j, \text{Immediate (16-bit)}$ $\backslash\backslash R_i = R_j \text{ AND Immediate Value}$
(32-bit unsigned extended)

11.OR R_i, R_j, R_k $\backslash\backslash R_i = R_j \text{ OR } R_k$.

12.ORI $R_i, R_j, \text{Immediate (16-bit)}$ $\backslash\backslash R_i = R_j \text{ OR Immediate Value}$
(32-bit unsigned extended)

13. HLT (Stops the execution).

- **OP-CODE for different commands:**

COMMAND	OP-CODE	Example Instruction (32-bit)
MOVE R_i, R_j	0000	0000 0xxx 0xxx 0000 0000000000000000
MOVE R_i , Immediate (16 bit)	0001	0001 0xxx 0000 0000 xxxxxxxxxxxxxxxxxx
LOAD $R_i, X(R_j)$	0010	0010 0xxx 0xxx 0000 xxxxxxxxxxxxxxxxxx
STORE $R_i, X(R_j)$	0011	0011 0000 0xxx 0xxx xxxxxxxxxxxxxxxxxx
ADD R_i, R_j, R_k	0100	0100 0xxx 0xxx 0xxx 0000000000000000
ADI R_i, R_j , Immediate (16 bit)	0101	0101 0xxx 0xxx 0000 xxxxxxxxxxxxxxxxxx
SUB R_i, R_j, R_k	0110	0110 0xxx 0xxx 0xxx 0000000000000000
SUI R_i, R_j , Immediate (16 bit)	0111	0111 0xxx 0xxx 0000 xxxxxxxxxxxxxxxxxx
AND R_i, R_j, R_k	1000	1000 0xxx 0xxx 0xxx 0000000000000000
ANI R_i, R_j , Immediate (16 bit)	1001	1001 0xxx 0xxx 0000 xxxxxxxxxxxxxxxxxx
OR R_i, R_j, R_k	1010	1010 0xxx 0xxx 0xxx 0000000000000000
ORI R_i, R_j , Immediate (16 bit)	1011	1011 0xxx 0xxx 0000 xxxxxxxxxxxxxxxxxx
HLT	1111	1111 0000 0000 0000 0000000000000000

* x means, the value can be taken 0 or 1 based on requirements.

=> The above table will be used to convert assembly code on LHS to machine code.