Arithmetic

- ADD r1,r2,r3 → r1 = r2 + r3
- SUB r1,r2,r3 → r1 = r2 - r3
- MUL r1,r2,r3 → r1 = r2 * r3
- DIV r1,r2,r3 → r1 = r2 / r3
- INC r1 → r1 = r1 + 1
- DEC r1 → r1 = r1 – 1

Logical

- AND r1,r2,r3 → r1 = r2 & r3
- OR r1,r2,r3 → r1 = r2 | r3
- XOR r1,r2,r3 → r1 = r2 ^ r3
- NOT r1,r2 → r1 = ~r2

Control Flow

- JMP addr → PC = addr
- BEQ r1,r2,addr → if(r1==r2) PC = addr
- BNE r1,r2,addr → if(r1!=r2) PC = addr
- CALL addr → stack[SP]=PC+1; SP--; PC=addr
- RET → SP++; PC=stack[SP]

Memory Access

- LD r1,addr → r1 = memory[addr]
- ST addr,r1 → memory[addr] = r1

Custom / Specialized

For all below operation only 9:0 bits of register will be consider

- FFT r1,r2 → memory[r1] = FFT(memory[r2])
- ENC r1,r2 → memory[r1] = Encrypt(memory[r2])
- DEC r1,r2 → memory[r1] = Decrypt(memory[r2])

PIPELINE STAGES AND CUSTOM OPERATION REMAINING…

Instruction	Opcode (binary 5-bit)
ADD	00000
SUB	00001
MUL	00010
DIV	00011
INC	00100
DEC	00101
AND	00110
OR	00111
XOR	01000
NOT	01001
JMP	01010
BEQ	01011
BNE	01100
CALL	01101
RET	01110
LD	01111
ST	10000
FFT	10001
ENC	10010
DEC (decrypt)	10011

Assembly Program

LD r2,addr //First store value in r1 from memory as value 23

LD r3,addr //r3 = 20

LABEL :

INC r3 //Increment content by 1

ADD r1,r2,r3 //add 23 + 21 and store into r1=44

BNE r2,r3 //if r2!=r3 then jump to LABEL (will loop for 3 times)

Memory Address	Instruction	Instruction in binary	Description
0000 0000 0	**LD r2,[100]**	01111 0010 0001100100	R2 = 23
0000 0000 1	LD r3,[101]	01111 0011 0001100101	R3 = 20
0000 0000 2	INC r3	00100 0011 0000000000	R3 = 20 + 1 = 21
0000 0000 3	ADD r1,r2,r3	00000 0001 0010 0011 00	R1 = r2 + r3
0000 0000 4	BNE r2,r3,0000 0000 2	01100 0010 0011 000010	Jump 2 location if(r2==r3)
0000 0000 5	ST r1,[102]	10000 0001 0001100110	
	END		

Memory location Register opcode Reserved

18:14	13:10	9:6	5:2	1:0
OPcode	Rd	Rs1	Rs2	reserved