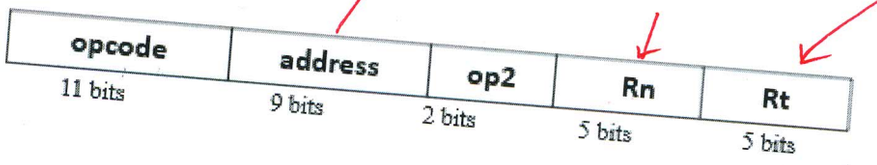
LEGv8 D-Format Instructions:

IF

constant offset . base Register

Destination Register



Translate the following LEGV8 assembly instruction into a machine instruction:

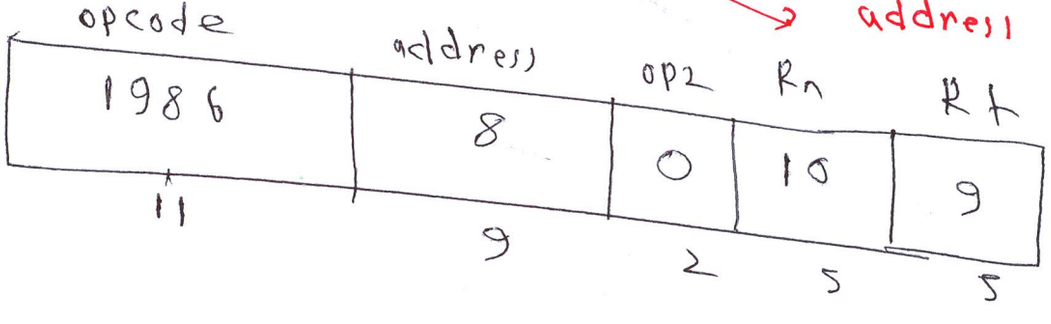
Destination register

base register

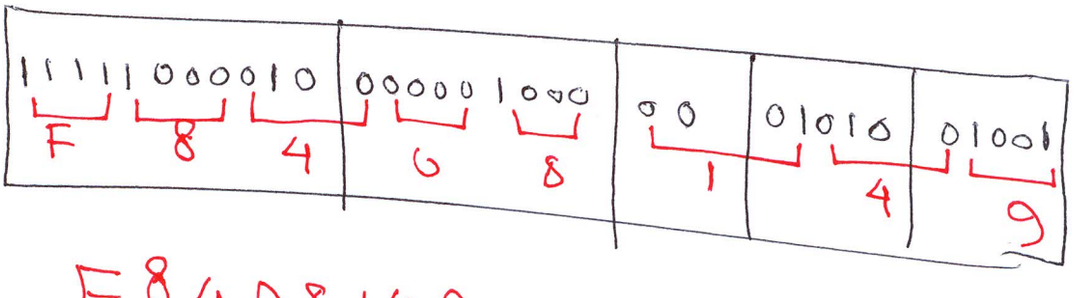
address

LDUR X9, [X10, #8]

ecimal. →



ary →



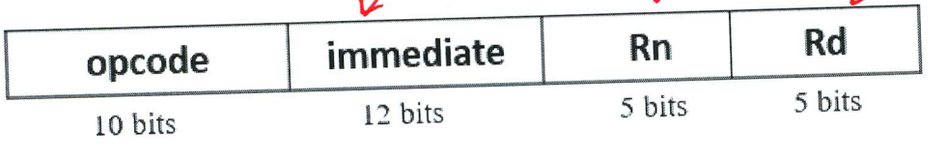
$F8408149_{16}$

LEGv8 I-Format Instructions

ADDI ✓
SUBI ✓

constant value ↓ Source register ↓ Destination register

I - Format



---

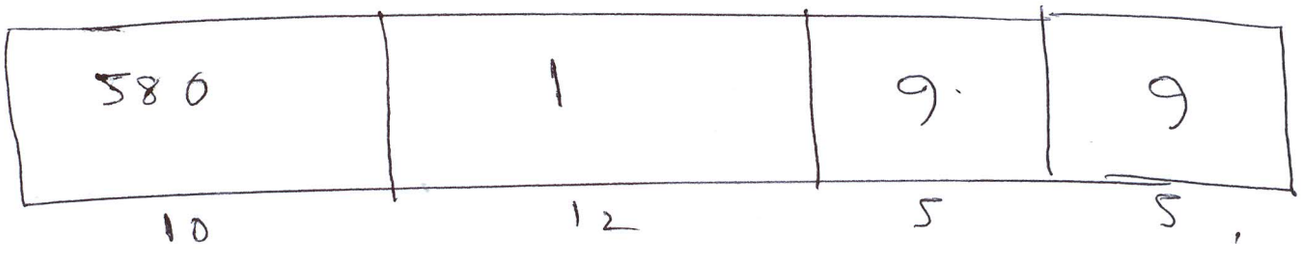
* Translate LEGv8 code to Machine code:

destination source

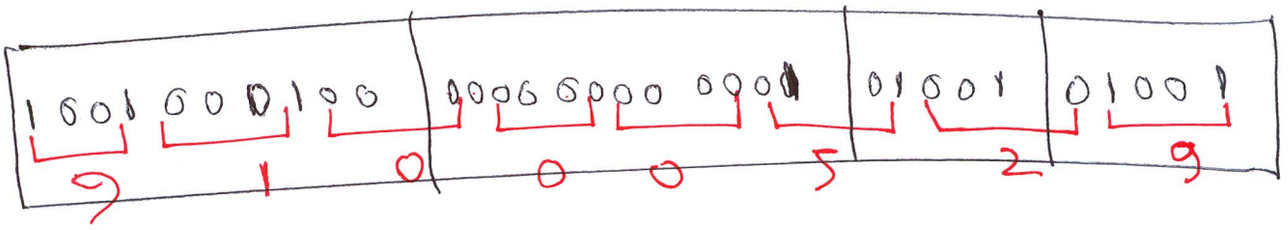
ADDI X9, X9, #1

opcode for
ADDI = 580
1001000100

Decimal



Binary.



Shift Operations: LSL
LSR

operand 2 (0) — shift amount — operand 1 — destination Register.

R-Format

opcode	Rm	shamt	Rn	Rd
11 bits	5 bits	6 bits	5 bits	5 bits

* Translate LSL X11, X19, #4 to Machine code

Opcode of LSL = 1691_{10}

11010011011_2

LSL X11, X19, #4 // X11 = X19 * 2^4.

X11 ↑ Destination Register (Rd)

X19 ↑ operand 1. (Rn)

#4 ↑ shamt

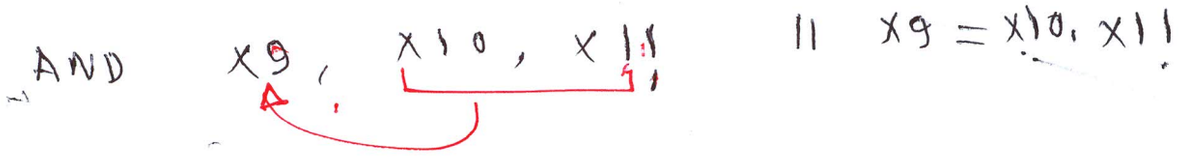
	opcode	Rm	shamt	Rn	Rd
Decimal	1691	0	4	19	11
	11	5	6	5	5
Binary	11010011011	00000	000100	10011	01011

$D3601268_{16}$

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

AND Operations

A logical bit-by-bit operation with two operands that calculates a 1only if there is a 1 in both operands.

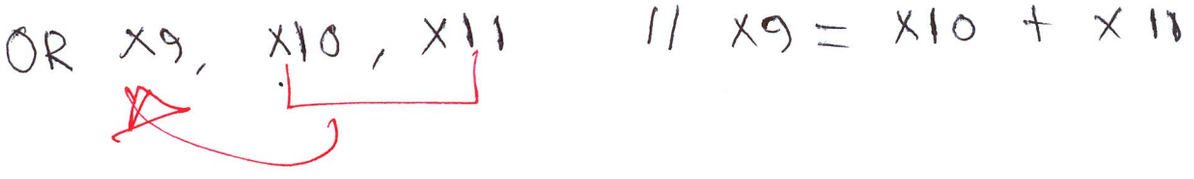


OR

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

OR Operations

A logical bit-by-bit operation with two operands that calculates a 1only if there is a 1 in either operands.



Branch to a labeled instruction if a condition is true – Otherwise, continue sequentially

CBZ register, L1

– if (register == 0) branch to instruction labeled L1;

– Example: CBZ X9, Else

CBNZ register, L1

– if (register != 0) branch to instruction labeled L1;

– Example: CBNZ X9, Else

B L1

– branch unconditionally to instruction labeled L1;

– Example: B Exit

	Signed numbers		Unsigned numbers	
Comparison	Instruction	CC Test	Instruction	CC Test
=	B.EQ	Z=1	B.EQ	Z=1
≠	B.NE	Z=0	B.NE	Z=0
<	B.LT	N!=V	B.LO	C=0
≤	B.LE	~(Z=0 & N=V)	B.LS	~(Z=0 & C=1)
>	B.GT	(Z=0 & N=V)	B.HI	(Z=0 & C=1)
≥	B.GE	N=V	B.HS	C=1

```
ADD  X0, X1, X2
CBZ  X0, L1
SUB  X2, X3, X5
L1 : Exit
```

conditional

```
ADD  X0, X1, X2
B L1
SUB  X2, X3, X5
L1 : Exit
```

unconditional

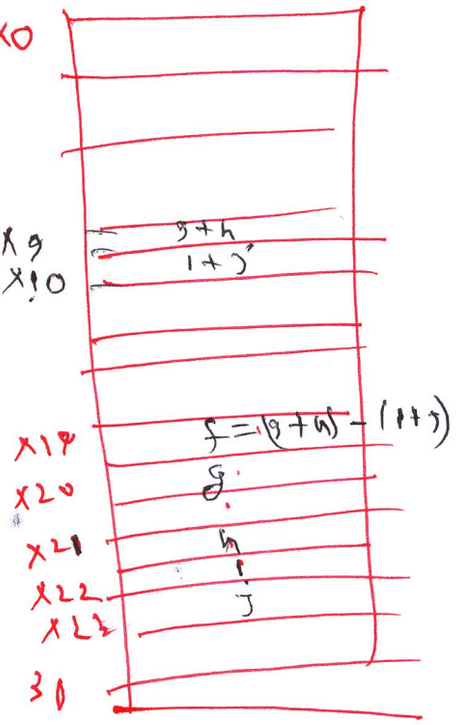
ADD MOV LSL
SUB LSR

Convert the following C++ code to LEGv8 Assembly code. Assume the variables f, g, h, i, and j correspond to five registers X19, X20, X21, X22, and X23.

f = (g + h) - (i + j);

X19, X20, X21, X22, X23
f g h i j

```
ADD  X9,  X20, X21   // X9 = g+h
ADD  X10, X22, X23   // X10 = i+j
SUB  X19, X9,  X10   // X9=(g+h)-(i+j)
                     // f
```



If statement

Convert the following C++ code to LEGv8 Assembly code. Assume the variables a and b correspond to registers X22 and X23.

```
if (a>b)
{
    a=a+1;
}
```

X22 , X23
↓ a ↓ b

if a <= b
Nothing happen
exit the program

```
CMP   X22, X23
      a    b
B.LE  Exit          check a <= b
ADDI  X22, X22, #1  // a = a+1
Exit:
```

Alternate way:

```
CMP   X22, X23
      a    b
B.GT  L1            a > b
B     Exit
L1: ADDI  X22, X22, #1   // a = a+1
Exit:
```