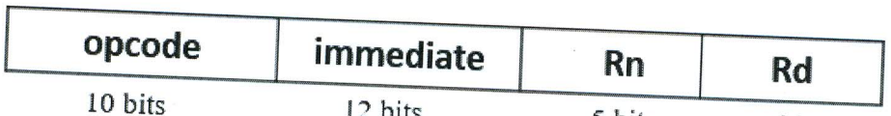
ADDI X9, X9, #1 // X9 = X9 + 1

SUBI X10, X7, #1 // X10 = X7 - 1

LEGv8 I-Format Instructions

Instruction format



Rn → source Register number

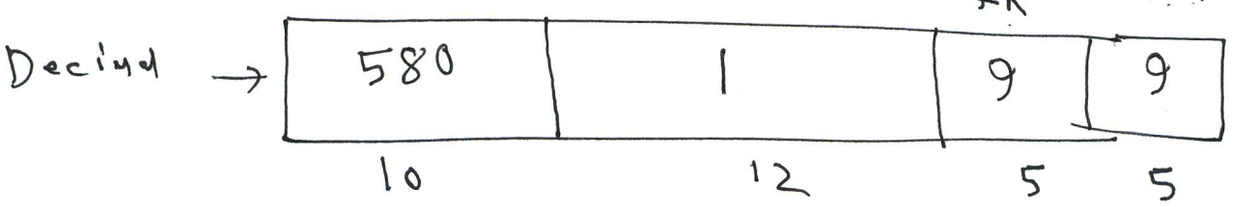
Rd → Destination Register number

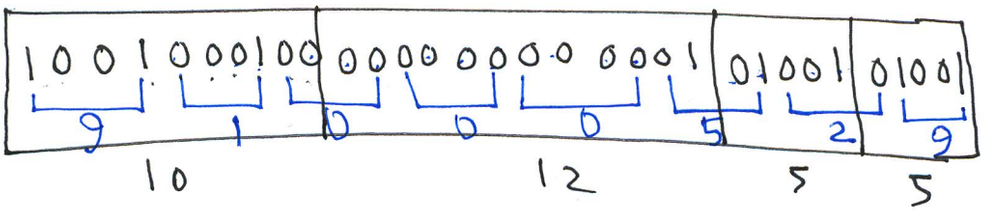
Example Translate following assembly instruction to machine instruction: ADDI X9, X9, #1

586

ADDI X9, X9, #1

Rd Rn immediate





Hex = 91000529 16

1691 Rd Rn Shamt

LSL X11, X19, #4 // X11 = X19 · 2^4

Shift Operations: LSR

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

R_M → operand 2 X (0)

shamt → shift amount

R_n → operand 1

Rd → Destination Register

1010 - A
1011 → B
1100 → C
1101 → D
1110 → E

1691 → 11.01.001.1.0 1.1.

19 → 10011

Example Translate LSL X11, X19, #4 to machine instruction and show the result in Hexadecimal.

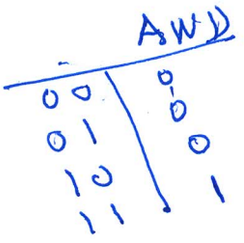
---

	opcode	R_m	shamt	R_n	Rd
Decimal →	1691	0	4	19	11
	11	5	6	5	5

Binary → 11010011011 | 00000 | 000100 | 10011 | 01011

D 3 6 0 4 C B

$D360126B_{16}$



AND Operations

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

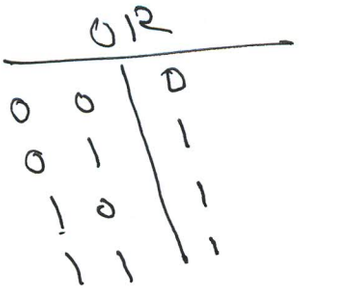
AND X9, X10, X11

X11 = 1100 0000

X10 = 0000 0000

X9 = 0000 0000

LSB



OR Operations

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

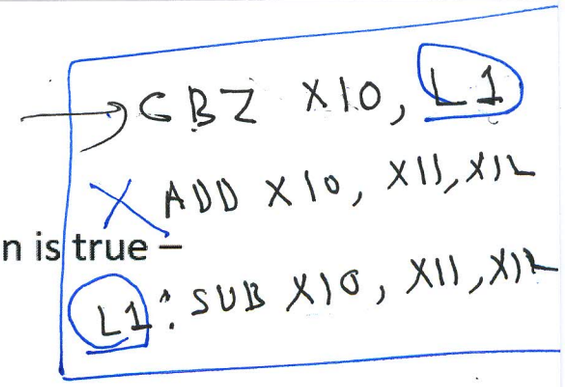
OR X9, X10, X11

X11 = 1100 0000

X10 = 0000 0000

X9 = 1100 0000

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

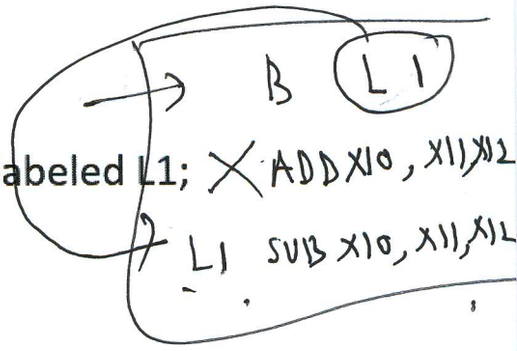
– if (register != 0) branch to instruction labeled L1;

– 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

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

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);
```

X9 X10 → c++

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

CMP → comparison

If statement

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

a b

X22 X23

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

```
0X2000:  CMP   X22, X23     // compare a and b
                 a    b      (a > b)
0X2004:  B.GT  L1
0X2008:  B     Exit
0X200C:  L1:   ADDI  X22, X22, #1   // a = a + 1
0X2010:  Exit:
```

Alternate solution

```
(*)      CMP   X22, X23
                 a    b
         B.LE  Exit
         ADDI  X22, X22, #1   // a = a+1
Exit:
```

Example

Assume variables a and b corresponds to registers X22 and X23

~~if (a = b)~~

```
if (a == b)
{
    b = 8*a ;
}
```

a → X22

b → X23

MUL

```
        CMP     X22, X23        // Compare a and b
        B.EQ
        B.NE    Exit            // go to exit if true
        LSL     X23, X22, #3    // b = 8*a
Exit:
```