Homework #1

1. LEGv8 coding

a) For the following C statement, write the corresponding LEGv8 assembly code. Assume that C variables f, g, h have already been placed in registers x8, x9, x10. Use a minimal number of LEGv8 assembly instructions

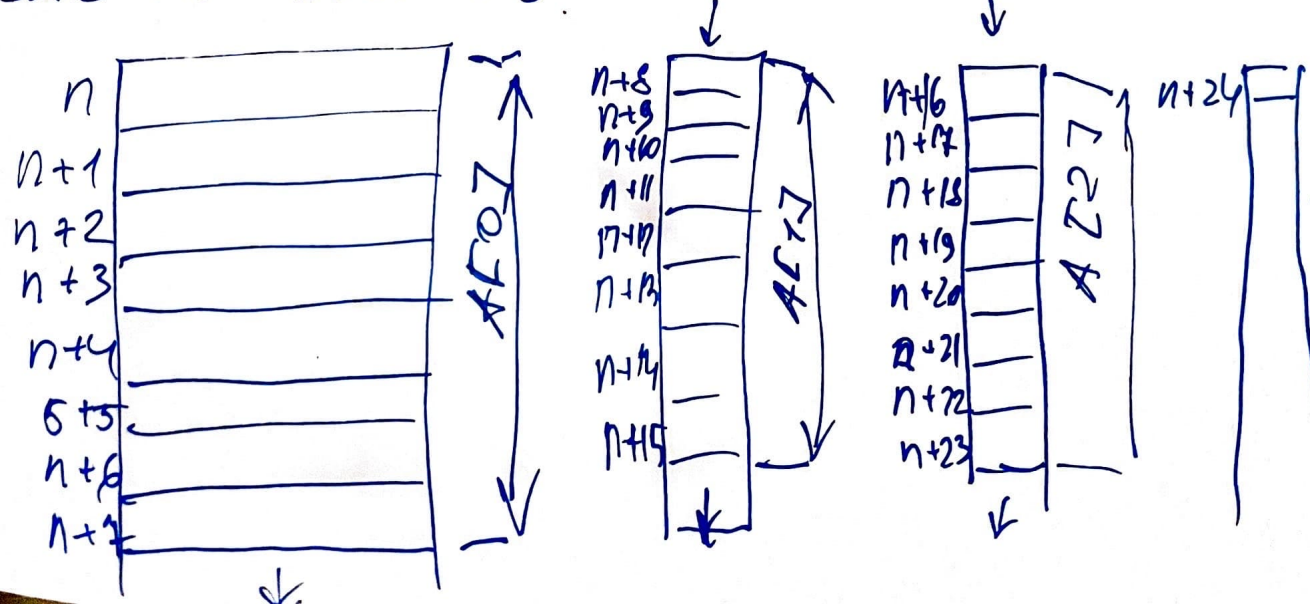
$$f = g - (h + 3)$$

```
ADDI X0, X10, #3   // X0 = X10 + 3
SUB  X8, X9, X0    // X8 = X9 - X0
```

b) For the following C statement, write a minimal sequence of LEGv8 assembly instructions that performs the identical operation. Assume D is in x5, and x7 is the base address of the integer array A.

$$D = A[3] >> 5$$

Need to use load, which allow to access data in memory



LDUR X5, [X7, #24]

takes value in X7, it adds 24 to it and it retrieve that value. it will lead us to begining address of 3rd element of array; X4 is register points to memory?

LSR

ADD X5, X1, #5 // >> means left Shift right; So what does it plus? Looks like we pointing to address and then making division of an address by 5.

[X7, #24] - here [ ] means we getting (extracting) value in the address #24 or only pointing to #24?

2. LEGv8 coding

ii) For the following C statement, write the corresponding LEGv8 assembly code. Assume that variables f, g, h, i, j are assigned to registers X0, X1, X2, X3, X4. Assume that the base address of the integer arrays A and B are in registers X8, X9

$$A[i+j] = B[5]$$

```
ADD X5, X3, X4 //X5 = i + j
LDR X10, [X9, #40] //loading B[5]; register X10 will load base address of B + offset of 40
~~CMP X8, X10~~
LSL X5, X5, #3 //X5 = 8 * (i+j)
```

b) For the following C statement, write the corresponding LEGv8 assembly code. Assume that the variables f, g, h, i, j are assigned to registers X0, X1, X2, X3, X4. Assume that the base address of the integer arrays A and B are in registers X8, X9.

$$A[j] = A[i] - B[3]$$

```
LDUR  X9, [X9, #24]   // X9 = B[3]
LSL   X18, X3, #3     // X8 = i * 8
LDUR  X8, [X8, #8]    // A[i]
SUM   X8, X8, X18     // X8 = base address of A
LDUR  X8, [X18, #0]   + offset of i*8
SUB   X15, X8, X9     // X15 = A[i] - B[3]

LSL   X4, X4, #3      // j * 8
ADD   X8, X4, X8      // A[j] + i*8
LDUR  X8, [X8, #0]    // A[j]
STUR  X11, [X15, #0]  // A[j] = A[i] - B[3]
```

3. Memory. a) Assume C is a character array with 20 elements stored in memory and it's starting memory address is in X9. What is the ~~element~~ memory address for element C[7]?

A Char data type takes 1 byte of memory. This array has 20 elements therefore size of char array is 20 bytes. Lets say we store element C[7] in register X7.

Address is X9 + 7

b) Assume A is an integer array with 20 elements stored in the memory and it's starting memory address is in X6. What's the memory address for ele A[9]

An integer array data type takes 4 bytes of memory. Therefore 32 bits each element.

Addres X6 + 36 // 36 = 9 · 4

c) Write at least 5 dif ways to assign 0 to X6.

```
MOV X6, X31 //X31 always hold value 0
MOV X6, #0
SUB X6, X6, X6
SUB X6, X31, X31
ADDI X6, #0, #0
MUL X6, XZR, X6
XOR X6, XZR, X31
ANDI X6, #0, XZR
```

4. Instruction

b) Following the instructions below consecutively and after each instruction write the value of destination register in decimal into the space provided.

```
ORRI X0, X31, #6  // X0 = 1   // 0|6
EORI X1, X0, #10  // X1 = 0   // (1 XOR 1) = 0
ADD  X2, X1, X1   // X2 = 0   // 0 + 0
LSR  X3, X2, #2   // X3 = 0   // 00000000 >> 2 = 00
SUB  X4, X2, X3   // X4 = 0   // 0 - 0
ANDI X5, X4, #7   // X5 = 7   // 0 + 7 = 7
```

4.b) For the following LEGB assembly, assume that registers X1, X6 contain values 24 and 256. Also assume that memory contains following values. What's the value of X0?

Address	Contents
256	50
260	200
264	550

```
SUBI X6, X6, #20  // X6 = 256 - 20 = 236
ADD  X6, X6, X1   // X6 = 236 + 24 = 260
LDUR X0, [X6, #4] // X0 = 260 + 4 = 264 = 550
```

! here we taking X6 that we got from calculations and adding 4, therefore we have [264] - it is address and need extract

5. Memory a) Show using tables below how the value 0X0012FF45F3AB440C (MSB ... LSB) would be arranged in memory of little Indian and a big-ending. Assume the data are stored starting at 4.

Big endian

address	Data
3	
4	0012FF45
5	F3AB440C
6	
7	
8	
9	
10	
11	
12	

Little endian

address	Data
3	C04 4BA3F
4	~~F3AB440C~~
5	~~0012FF45~~ 54FF2100
6	
7	
8	
9	
10	
11	
12	

Address	Contents
0X0000000F	0X00
0X0.... 10	0x00
11	0x00
12	0xAA
13	0xCC
14	0x88
15	0x99
16	0x11
17	0x23
18 FF	0x45
19 EE	0x00
1A ~~20~~ 00	0xFF
1B A8	0x00
1C 0C	0xAA
1D 00	0xBB
1E 33	0x1C
1F 52	0x2F
20	0x55
21	0x44
22	0xFB

In Big endian MSB stored in the lowest address, while In Little endian LSB stored in the lowest address.

b) Based on the given memory map → what is the result of the LEGv8 instruction LDUR X1, [X3, #3], if the machine is in little-endian byte order, where X3 = 0x00000010 (i.e what is the value of X1)

[X3, #3] = 00000010 + 3 = 00000013 which is equal to 0xCC, then 88, 99, 11; but in Little endian less significan bit at the lowest memory address therefore (up side down)

X1 = 0x11 99 88 CC

c) Show on the memory map the result of the following instruction.

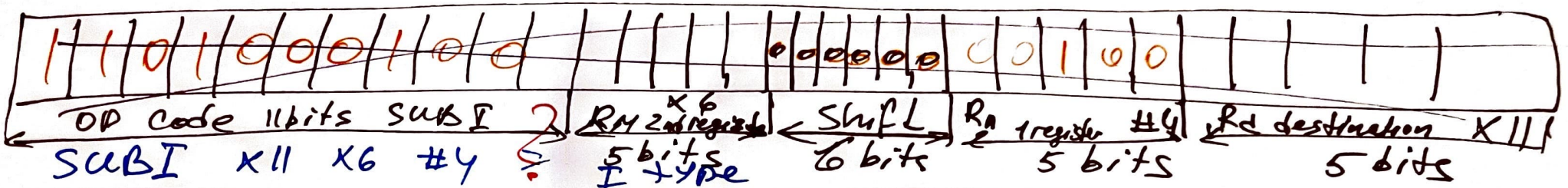
STUR X2, [X0, #-4]

where X0 = 0x000000001C, X2 has the value of 0xFFFE00A80C0D3352 (i.e. show the content changes of the corresponding memory cells). Assume that this is big endian machine.

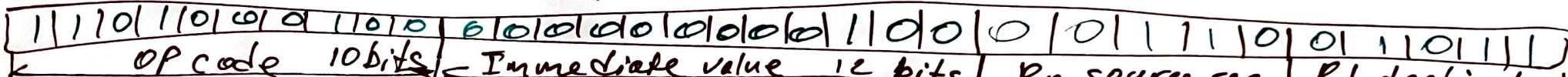
X0 = 0x000000001C − 4 = 0x0000000018

and in this memory address store value X2

Please result on map.

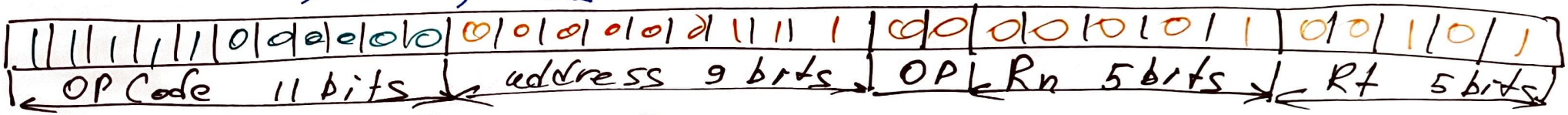


SUBI X11 X6 #4 ?

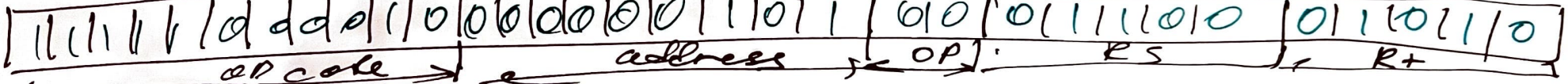


STUR X5, [X1, #8]

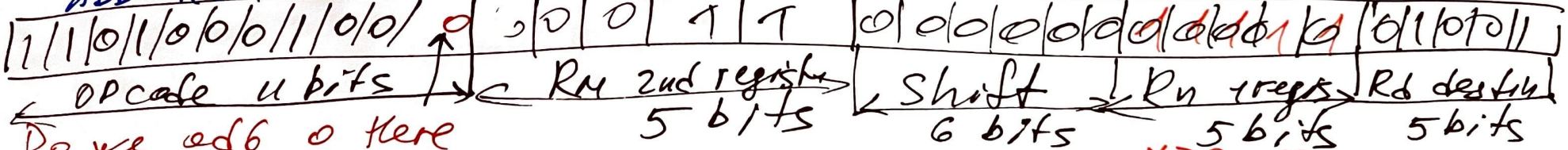
D-TYPE (for arrays)



LDUR X10, [X12, #5]

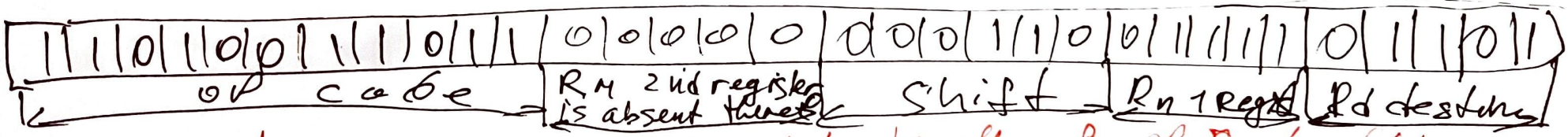


ADD X9, XZR, X3



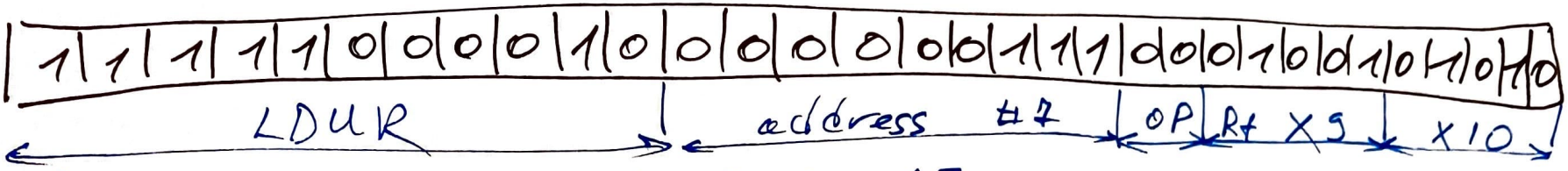
? Do we add 0 here

LSL X13, X15, #6

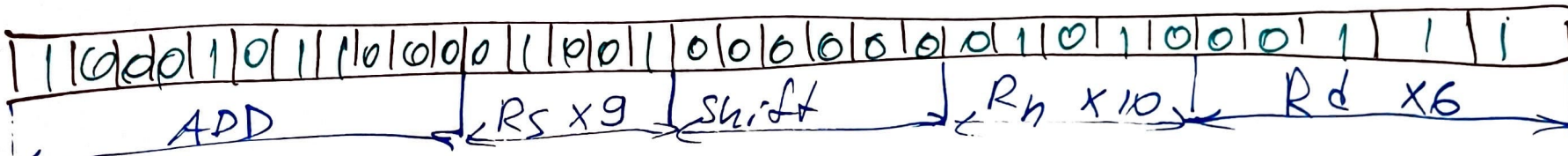


? Do we have to remember bit length of OP Instructions

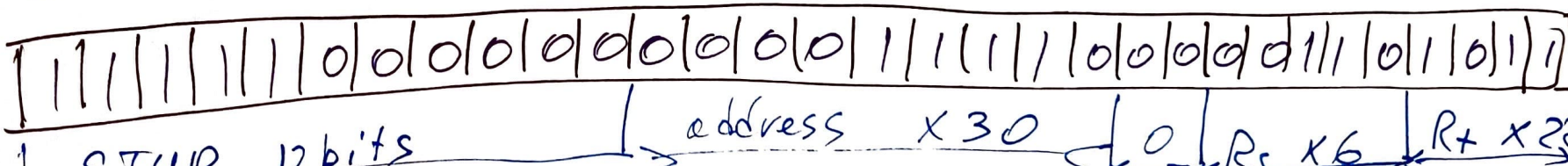
7. Instruction decoding provide the instruction type and assembly language instruction for the following binary values



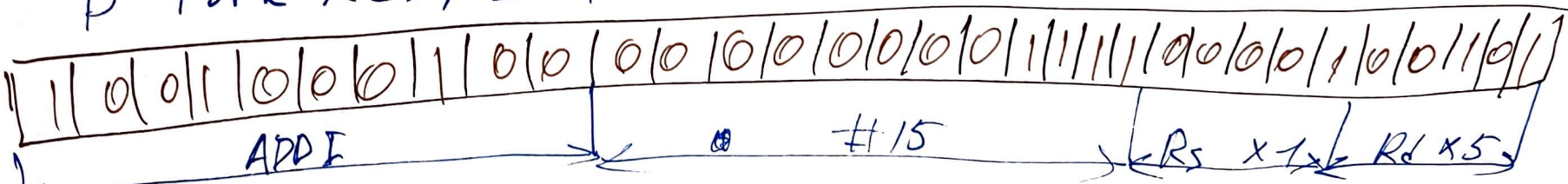
D-TYPE: LDUR X10, [X9, #7]



I-TYPE ~~X6~~, ADD X6, X10, X9



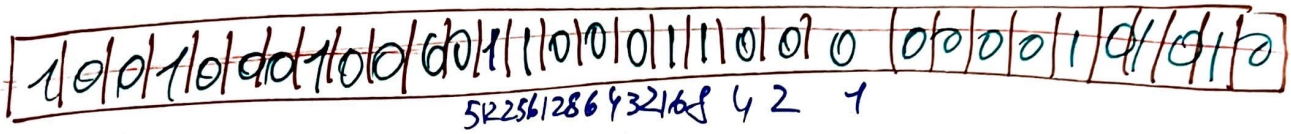
D-TYPE X23, [X6, #30]

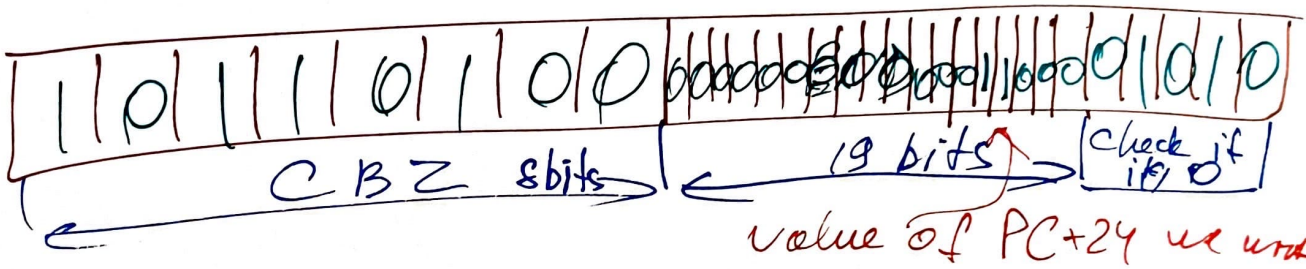


I-TYPE ADDI X5, X1, #15

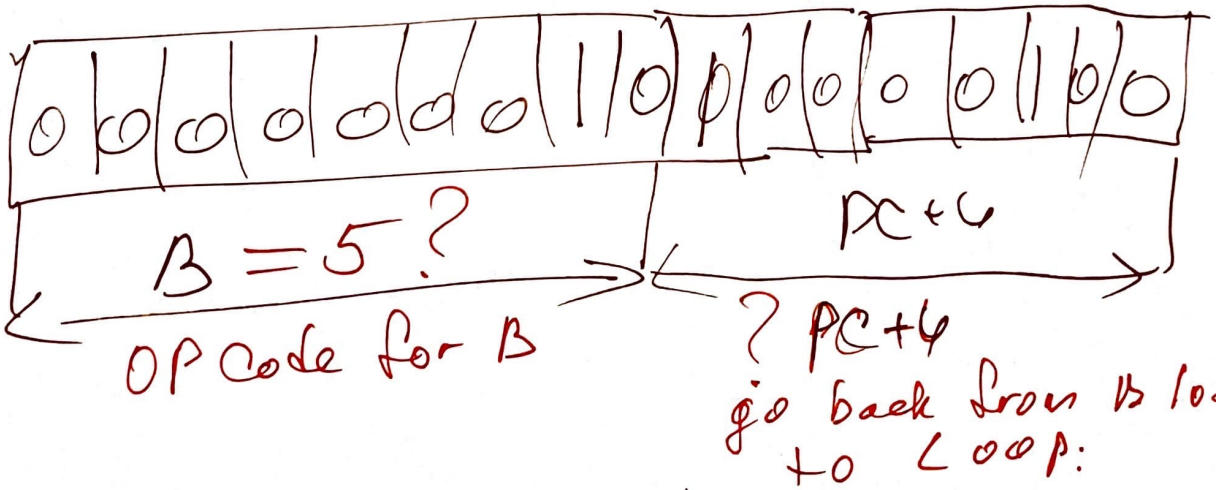
9. For the following program, represent CBZ and B instructions in binary.

```
   ADDI X10, X1, #792   <- PC   (792 = 768 + 24)
LOOP: CBZ X10, END      <- PC+4
      LDUR X11, [X10, #0]  <- PC+8
      ADD X0, X0, X11      <- PC+12
      SUBI X10, X10, #8    <- PC+16
      SUB X10, X10, X1     <- PC+20
END:  B LOOP               <- PC+24
```





B-TYPE



10. Write a segment of LEGV8 instructions to do the following: set X5 to 20 if the contents of X0 is an even number and set X6 to 21 otherwise. Do not use division instruction in your code. Instead, you use and AND instruction to test to test if a number is even or odd. Comment your assembly code.

```
AND X0, X0, X0  // X0 = 0 & 0 or 1 & 1
CBZ X0, else    // if X0 = 0, jump to else branch
MOV X6, #21     // if X0 != 0 then X6 = 21
  B exit        // then jump to exit ? to put END
else MOV X5, #20 // if X0 = 0, X5 = 20
exit END)       // end program
```

11. Write the corresponding LEGV8 code for the following fragment of C code.

```
for (int i = 2; i < 50; i++)
    C[i-j] = C[i] - C[i+1] * 15;
```

Assume that the index i is in register X0, C is an integer array and the base address of C is in X1. Comment your assembly code. Note do not use multiply instructions. Use some other way to multiply by 15. Hint: which other instructions multiply? How many instructions are executed? How many data memory references have been made?

```
Loop  ADDI X0, XZR, #2 // i = 0 + 2
Loop  SUBI X4, X0, #50 // i = 0 - 50
      CBZ X4, Exit // if i==0, exit
      LSL X5, X0, #4
```

? #4 is $2^4 = 16$ which is address of second element

? what it gets us (offset)

```
      ADD X1, X1, X5 // x1 is base of the
```

array + we adding offset to second element ?
Does it gives address of first element ?

```
      LDUR X8, [X1, #8] // loading 2 into X8
```

? can it be x1 here

```
      LDUR X6, [X1, #8] // base of array +1
```

? #8 because of 1

```
      ADDI X3, X6, #120 // C[i+1]. 15 ?

      SUB X22, X8, X3 // C[i] - C[i+1]. 4
      LOAD X25, [X8, #-8] // C[i-1]
      STUR X25, [X22, #0] // C[i-1]=C[i]-C[i+1]
      ADD X0, X0, #1 // i = i + 1
      B Loop // jum up to loop
exit: // end
```