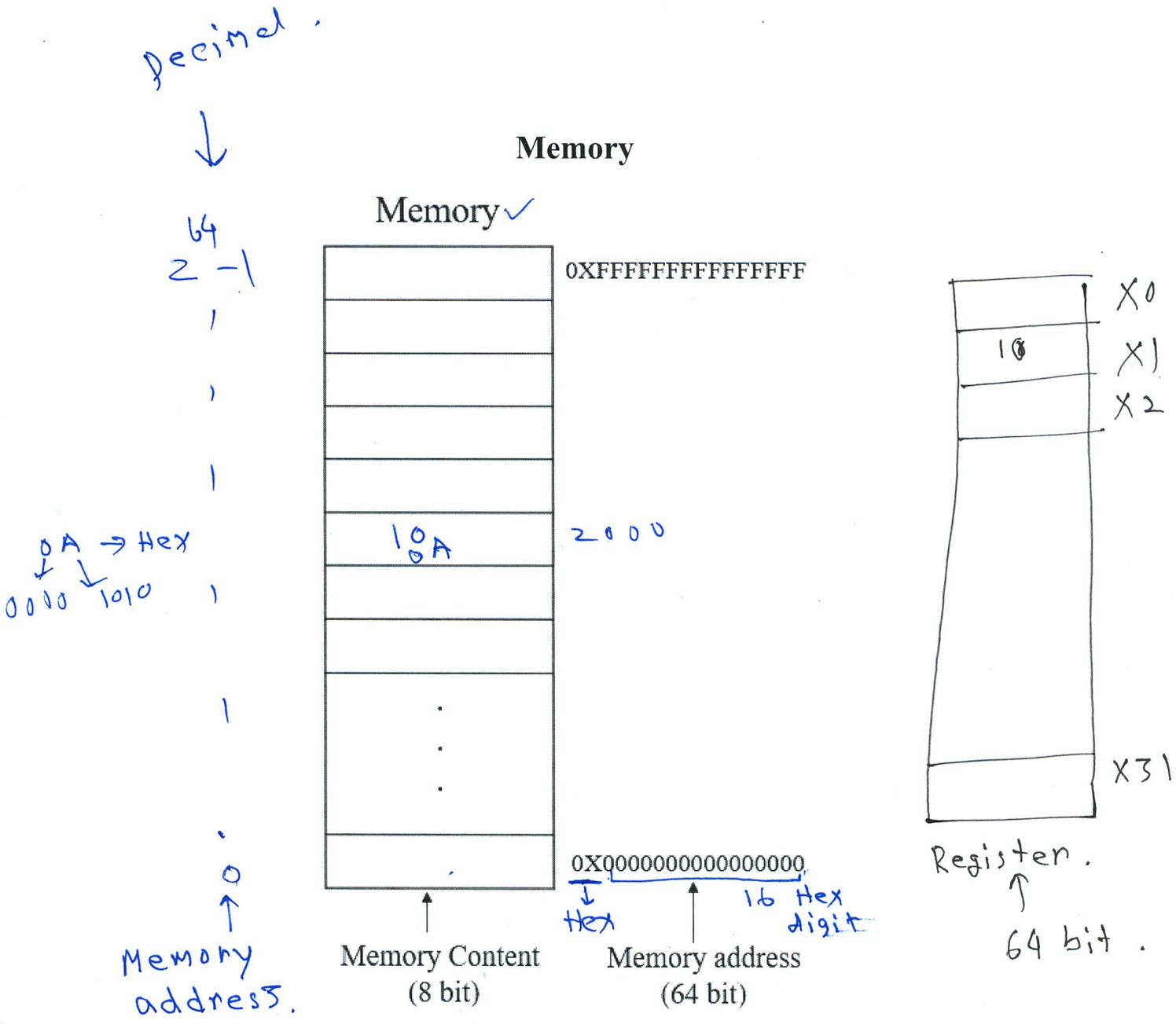
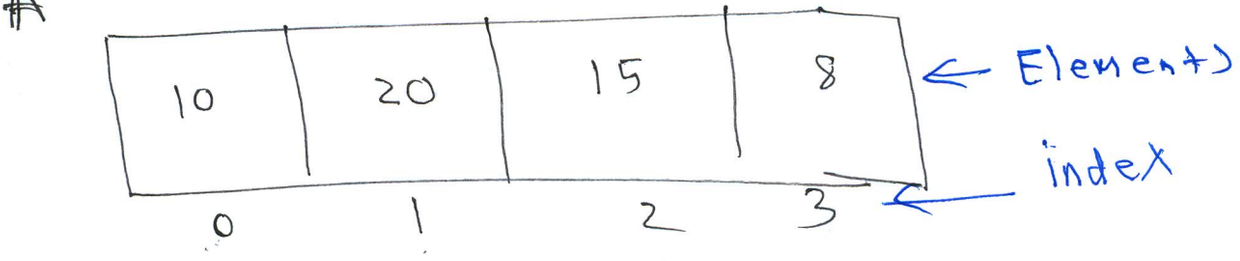


Problem: Assume that A is an array of 4 integer type elements (10, 20, 15, 8). Each element is 64 bit (doubleword). How does the array elements contained on the memory ? The base address of the array A is 0X0000000000000000.

int A = {10, 20, 15, 8};



8bit = 1 byte = 2 Hex digit ↓Hex 0A = 8

A[0] = 10 → 0000 0000 0000 00 0A

A[1] = 20 → 0000 0000 0000 00 14

A[2] = 15 → 0000 0000 0000 000F

A[3] = 8 → 0000 0000 0000 0008

↓ LSD

Address of A[0] = Base address + 0 * 8 = 0X0000 0000 0000 0000

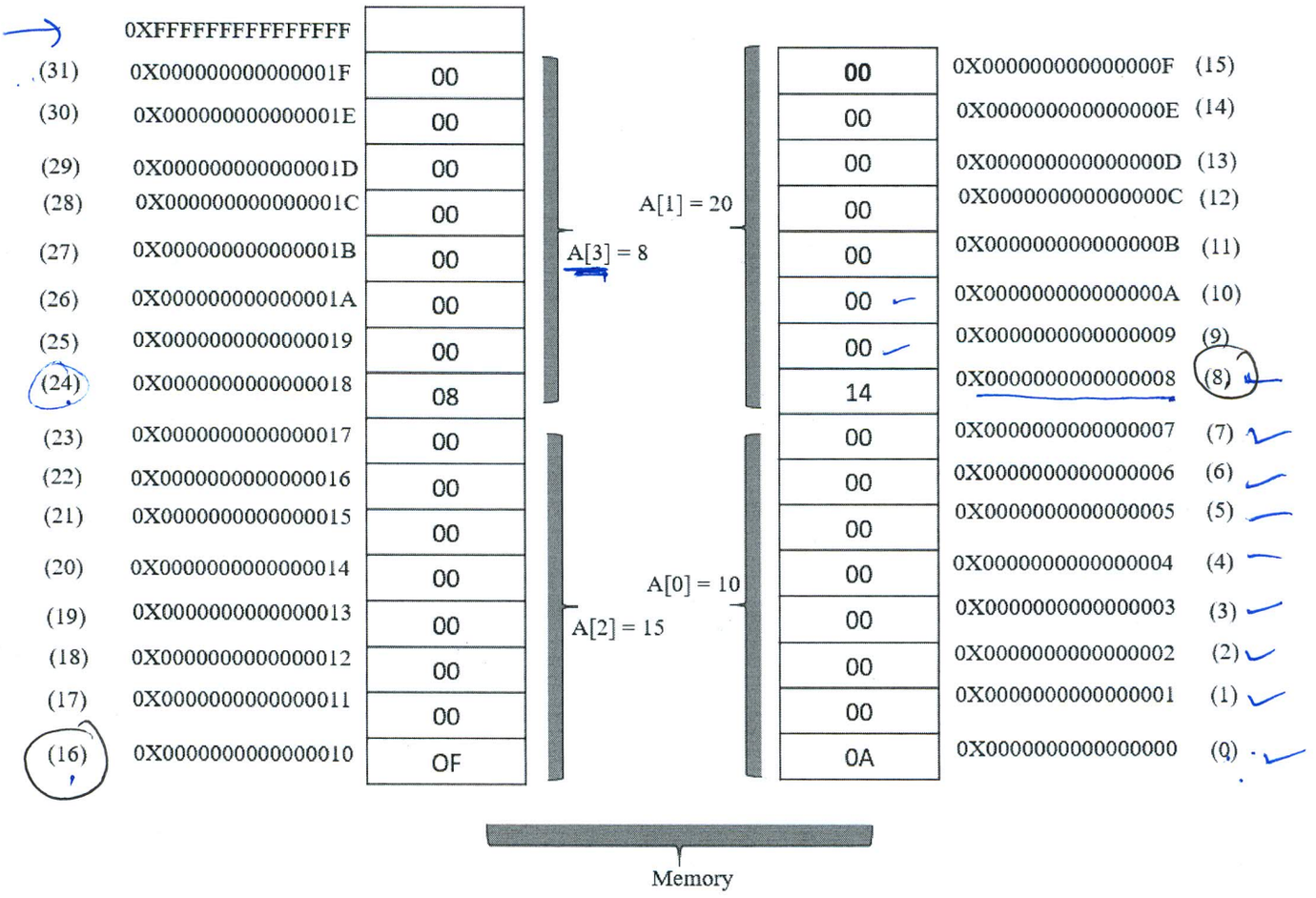
Address of A[1] = Base address + 1 * 8 = 0X0000 0000 0000 0008

Address of A[2] = Base address + 2 * 8 = 0X0000 0000 0000 0010

Address of A[i] = Base address + i * 8

64

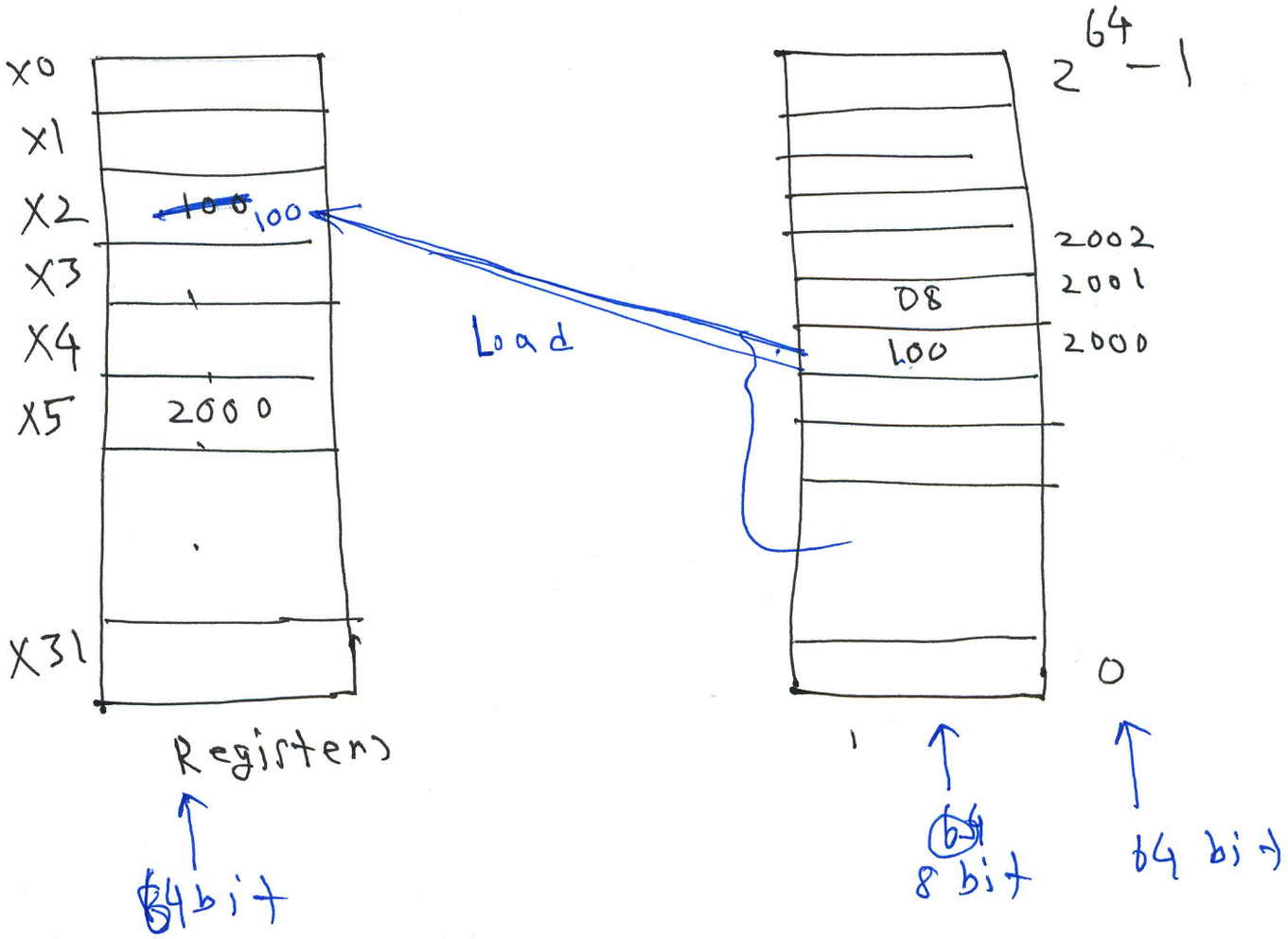
2 − 1 →



Load/Store Instruction

The ARM CPU allows direct access to all locations in the memory, but they are done with specific instructions. Since these instructions either load the register with data from memory or store the data in the register to the memory, the are called load/store instructions.

Assume, ~~X2 = 100~~ X5 = 2000 Memory.



LDUR X2, [X5, #0] ;

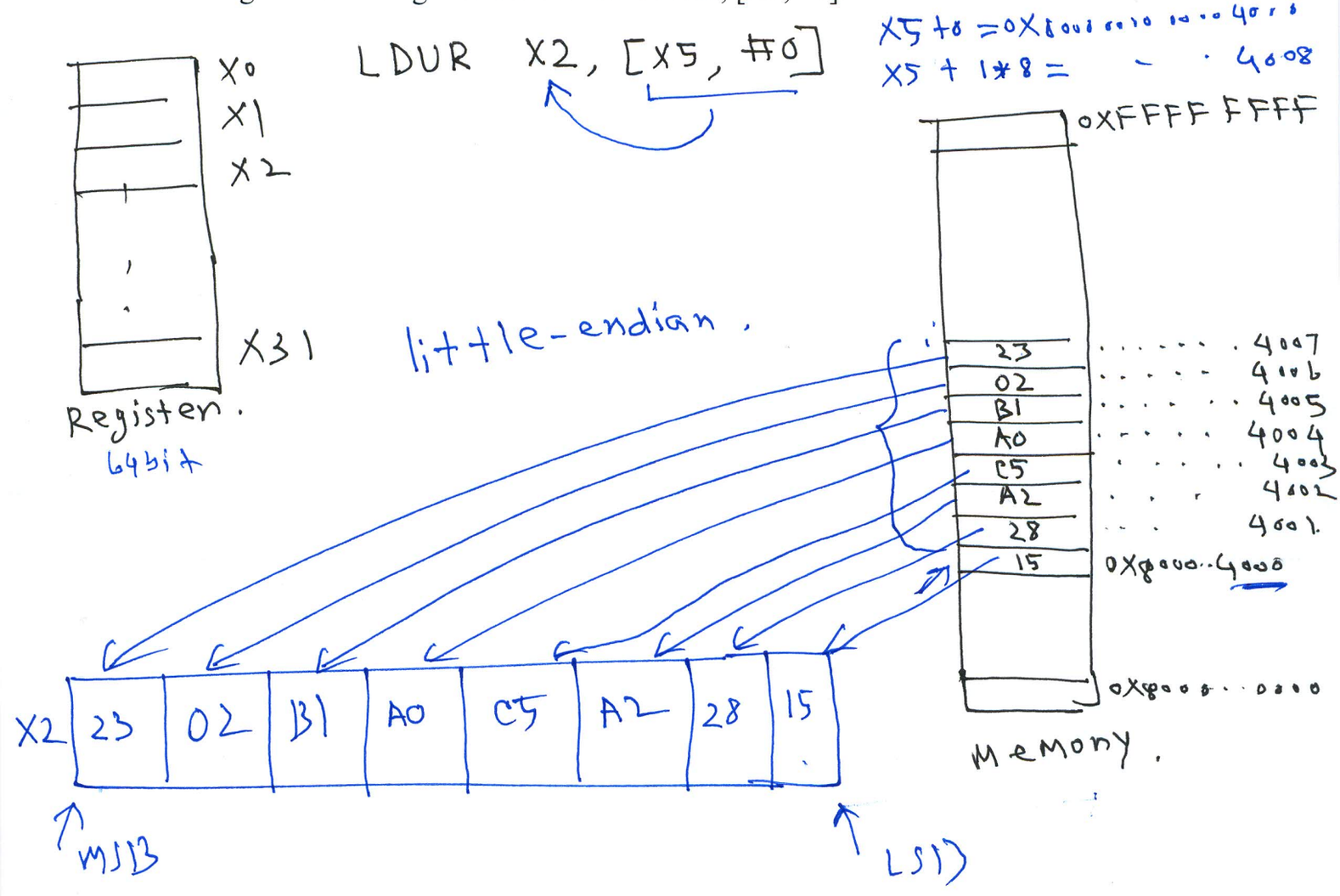
→ LDUR X2, [X5, #0] memory [X5 + 0]
= [2000 + 0]
= [2000]

Register

Load Instruction

- LDUR X2, [X5, #0] instruction copies the content of the memory locations pointed by X5 + 0 into register X2.
- Since the X2 register is a 64-bit wide, it expects a 64-bit operand in the range of 0X0000000000000000 to 0XFFFFFFFFFFFFFFFF.

Problem: Assume that X5 = 0X8000000000004000 and locations 0X8000000000004000 through 0X8000000000004007 contain 0X15, 0X28, 0XA2, 0XC5, 0XA0, 0XB1, 0X02, and 0X23, respectively. What will happen to X2 after running the following instruction: LDUR X2, [X5, #0].



address A[1] = Base address + 1 * 8

= X22 + 8

Example: Convert the following C++ code to LEGv8 Assembly code. Assume A is an Array of 10 doublewords, variable g and h are associated with registers X20 and X21, and base address of the array A is in X22.

(1) g = h + A [1];

(2) g = h + A [8];

X22 → A. X20 → g, X21 → h.

1. g = h + A [1] → c++

```
LDUR  X9, [X22, #8]   // X9 = A[1]
ADD   X20, X9, X21    // g = h + A[1]
```

2000 + 8 = 2008