Senni Tan 400196392 T02

1) 2.14
   STUR X9, [X10, #32]

   Opcode: 1984 → 11111000000

   Address: 32 → 000100000

   Op2: 00

   Rn: 10 → 01010

   Rt: 9 → 01001

   0b 11111000000 000100000 00 0101001001

   0b 1111 1000 0000 0010 0000 0001 0100 1001

   0x F 8 0 2 0 1 4 9

   Answer: 0xF8020149

2) 2.16

   Op = 0x7c2 = 1986 = 11111000010 (LDUR)

   Rn = 12 = 01100

   Rt = 3 = 00011

   Const = 0x4 = 4

   Address = 4 * 8 = 32 = 000100000

   Type: D – type

   Assembly: LDUR X3, [X12, #32]

   Binary representation: 0b 11111000010 000100000 00 01100 00011

3) 2.17.2
   Log2(128) = 7
   Rn: 5 bits → 7 bits
   Rd: 5 bits → 7 bits
   △bits = 4
   Opcode: 10 – 4 = 6 bits
   Immediate: 12 bits
   ∴ Rn and Rd increased to 7 bits, opcode decreased to 6 bits, immediate remains 12 bits.

4) 2.4

```
ADDI X11, X9, #8 // X11 = &A[f+1]
LDUR X9, [X11, #0] // X9 = A[f+1]
ADD X9, X9, X0 // X9 = A[f] + A[f+1]
STUR X9, [X10, #0] // B[g] = A[f] + A[f+1]
```

Answer: B[g] = A[f] + A[f+1]

5) 2.8

```
LSL X9, X3, #3  // X9 = i * 8
ADD X9, X6, X9 // X9 = &A[i]
LDUR X0, [X9, #0] // f = A[i]
LSL X10, X4, #3 // X10 = j * 8
ADD X10, X6, X10 // X10 = &A[j]
LDUR X1, [X10, #0] // g = A[j]
ADD X2, X0, X1 // h = f + g = A[i] + A[j]
STUR X2, [X7, #64] // B[8] = h = A[i] + A[j]
```

6) 2.22

X0 is greater than 0

So jump to ELSE

Bitwise OR on 0 and 2, then store in X1

Therefore, The value of X1 after instructions is 2.

7) 2.25.4

X0's final value depends on X1. In this question, we are not given the X1 value. Assume X1 value is N. Then the LOOP will execute N + 1 times. And in each loop, X0 increases by 2. Therefore, the X0's final value is 2*(N + 1).

8) 2.25.5

```
While(X1 >= 0)
{
    X1 = X1 – 1;
    X0 = X0 + 2;
}
```

9) 2.28

```
int i = 0;
int result = 0;
while(i < 100){
    result = result + MemArray[0];
    MemArray = MemArray + 1;
    i = i + 1;
}
```

10) 2.36.2
    0x88

11)
    0x 8877665544332211 =
    0b 1000 1000 0111 0111
       0110 0110 0101 0101
       0100 0100 0011 0011
       0010 0010 0001 0001

    4 bits/digit, so shift left by 4 for a digit position.

    Code:

```
    ADDI X10, XZR, #8 // x10 = 8
LOOP:  ADDI X3, X10, #0 // x3 = x3 + x10
    LSL X3, X3, #4 // x3 = x3 * 16
    ADDI X3, X10, #0 // x3 = x3 + x10
    LSL X3, X3, #4 // x3 = x3 * 16
    SUBI X10, X10, #1 // x10 = x10 – 1
    CMPI    X10, #0
    B.GT    LOOP
    LSR  X3, X3, #4 // the result is 16 times greater, therefore x3 = x3/16
```

12) 3.23

63 = 111111

0.25 = 1/4 = .01

→ 111111.01 = 1.1111101 * 10^5

Exp = 127 + 5 = 132 = 10000100

Float: 0 10000100 11111010000000000000000