1. lui $t0, 3840

lb $s0, 2($t0). $(47)_{16} = (0100\ 0111)_2$

content of $s0: 0000 0000 0000 0000 0000 0000 0100 0111

2.

```
FACT: addi $sp, $sp, 8  -8
      sw $ra, 4($sp)
      sw $a0, 0($sp)
      add $s0, $0, $a0
      slti $t0, $a0, 2
      beq $t0, $0, L1
      mul $v0, $s0, $v0   addi $v0, $0, 1
      addi $sp, $sp, -8   8.
      jr $ra
L1:   addi $a0, $a0, -1
      jal FACT
      addi $v0, $0, 1
      lw $a0, 0 4($sp)
      lw $ra, 4 0($sp)
      addi $sp, $sp, -8   8   = mul $v0, $a0, $v0
      jr $ra
```

(10) 16.

0001 0000

3. 0000 00 | 01 11 | 0 | 10 | 0 | 01 00 | 000 00 | 0 00 | 0

t3 t2 t1 32+2

0 11 10 9 34

then funct is sub.

1) ⇒ sub $t1, $t3, $t2

2) R-type instruction

4. lw I-type 23 H

$s1 17, $s2 18.

op $s1 $s2 -32

23H 17 18 -32.

1)

⇒ 100011 10010 10001 11111 11111 100000

2) I-type

5. 32 - 5 bits 128 - 7 bits.

1) for R-type.

rs, rt, rd now needs 3×7 = 21 bits.

shamt, op, funct remain 5+6+6 = 17 bits.

so it needs 21+17 = 38 bits.

2) for I-type

rs, rt needs 2×7 = 14 bits.

op, constant or address needs 16+6 = 22 bits.

14+22 = 36 bits.

6. 64 - 6 bits.

1) op (6 bits) rs (6 bits) rt (6 bits) constant or address (14 bits).

$-2^{13} \sim 2^{13}-1$

When 32 register, relative address has range $0 \sim 2^{16}$

64 register only has range $0 \sim 2^{14}$ relative address

2) no impact.

jr: R-type

7.

0x1000F400 : 000000 00000 01000 0110 1 00000 10 10

0x1000F404 : 000101 0110 1 00000 00000 00000 000001

0x1000F408 : 000010 10000 0000 0000 1111 0100 0001 10

0x1000F40C : 001000 10011 10011 00000000000000 10

0x1000F410 : 001001 01010 01010 00000000000000 01

0x1000F414 : 000010 10000 0000 0000 1111 0100 0000 00

0x1000F418 : ...

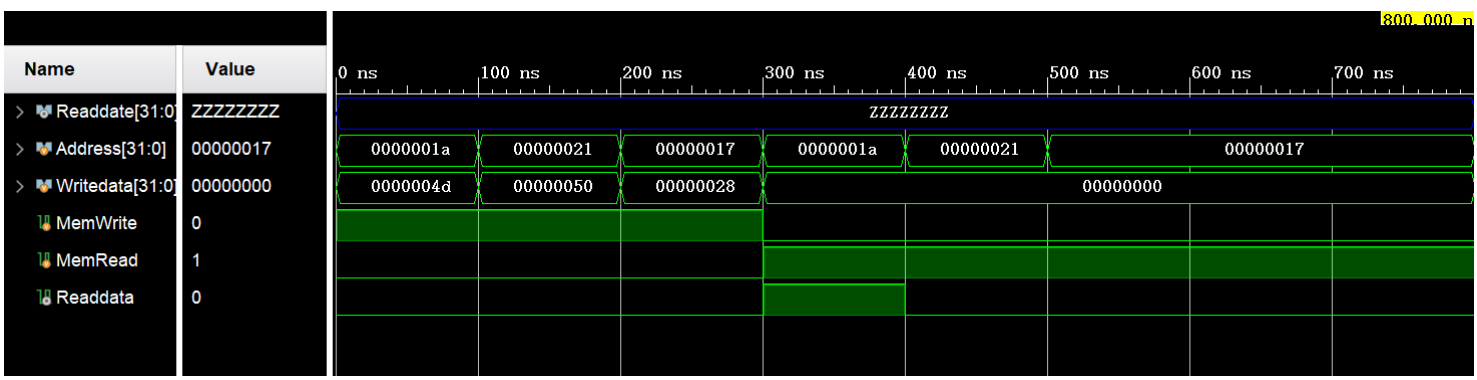
8.

```
module memory(Readdata, Address, Writedata, MemWrite, MemRead );
    parameter width=32;
    parameter addr_width=32;


    output[width-1:0] Readdata;
    input[width-1:0] Writedata;
    input[addr_width-1:0] Address;
    input MemWrite,MemRead;

    reg[width-1:0] Readdata;
    reg[width-1:0] memory[100:0];

    always@(posedge MemWrite or posedge MemRead  or posedge Address)begin
        if(MemWrite)begin
            memory[Address][31:0]=Writedata[31:0];
            Readdata=0;
        end
        if (MemRead)begin
           Readdata[31:0]=memory[Address][31:0];
        end
    end
endmodule
```



9.

```
23 module alu(Ainvert,Bnegate,Result,Operation,a,b,clk,Overflow);
24 input Ainvert,Bnegate,clk;
25 input[1:0]Operation;
26 input[31:0]a,b;
27 output[31:0]Result;
28 output Overflow;
29 reg[32:0] op_overflow;
30 reg[31:0] Result;
31 reg[31:0] A,B;
32 reg Overflow;
33 always@(posedge clk)begin
34     Result='bz;
35     if(Operation==0)begin
36         Result=a&b;
37     end
38     if(Operation==1)begin
39         Result=a|b;
40     end
41     if(Operation==2)begin
42         if(a<b) Result=1;
43         else Result=0;
44     end
45     if(Operation==3)begin
46         A=a;
47         B=b;
48         if(Ainvert) A=-a;
49         if(Bnegate) B=-b;
50         op_overflow=A+B;
51         Result=op_overflow;
52         if(op_overflow[32]==0)  Overflow=0;
53         else Overflow=1;
```

end
end
end
end module

10.

Executable File Header		
	Text size	0x440.
	Data size	0x90.
Text Segment	Adress	Instruction
	0x00400000	lui $at, 0x10000000
	0x00400004	ori $a0, $at, 0x10000000
	...	...
	0x00400140	sw $a0, 8040($gp)
	0x00400144	jmp 0x04002C0
	...	...
	0x004002C0	jal 0x04000000
	...	...
Data Segment	Address	
	0x10000000	(X)
	...	...
	0x10000040	(Y)
	...	...