Homework 14

Problem 1: Y86

0x000:
0x000: 30f400010000
[1] : 30f500010000
0x00c: 7020000000
0x014:
0x014:
0x014: 000b0000
0x018: [2]
0x01c: [3]
0x020: 30f214000000
0x026: a02f
0x028: 802e000000
0x02d: 00
0x02e: a05f
0x030: 2045
0x032: 501508000000
0x038: 30f203000000
0x03e: 6222
0x040: [5]
0x045: 506100000000
0x04b: 2060
0x04d: 30f304000000
0x053: 6031
0x055: 30f3ffffffff
0x05b: 6032
0x05d: 7445000000
0x062: b05f
0x064: 90
0x100:
0x100:
	# Execution begins at address 0
	.pos 0
	init: irmovl Stack, %esp
	irmovl Stack, %ebp
	jmp Main
	# Array of 4 elements
	.align 4
	array:
	.long 0x00000b00
	.long 0x0fff6320
	.long 0x12345678
	Main: [4]
	pushl %edx
	call Foo
	halt
	Foo: pushl %ebp
	rrmovl %esp,%ebp
	mrmovl 8(%ebp),%ecx
	irmovl $3,%edx
	andl %edx,%edx
	je End
	Loop: mrmovl (%ecx),%esi
	rrmovl %esi,%eax
	irmovl $4,%ebx
	addl %ebx,%ecx
	irmovl $-1,%ebx
	addl %ebx,%edx
	jne Loop
	End: [6]
	ret
	.pos 0x100
	Stack:

1) Fill in the blanks according to the Y86 assembly and object code. (1'*6 = 6')

2) Fill in the value of the registers after the above program has halted. (1' * 7 = 7')

%eax: [1] %ebx: [2] %ecx: [3] %edx: [4]

%esi: [5] %edi: -- %ebp: [6] %esp: [7]

Problem 2: Cache

Consider a 32-bit machine with a 4-way set associative cache. There are 16 sets.

Each block is 32 bytes. Answer the following questions:

1. please fill the following blanks

Cache size: [1] bytes

Field	Length(bit)
Total	32
Tag	[2]
Set	[3]
Offset	[4]

2. Assume the cache is initially empty. There are several memory accesses in order.

Fill the second table and compute the miss rate.

Order	Address	Set	Miss or not(Yes/No)
1	0xe166	[1]	[2]
2	0xe17e	[3]	[4]
3	0xe15a	[5]	[6]
4	0x9d60	[7]	[8]
5	0x9d5f	[9]	[10]
6	0xe142	[11]	[12]

Miss rate: [13]

3. Assume the following code is executed on this machine. You should also assume the following:

(1)sizeof(int) = 4

(2)the starting address of array a is 0x100

(3)variable i, j, t are in registers

```
#define M 4
#define N 8
int a[M][N];
int i, j;
for (i = 0; i < M - 1; i++) {
        int t = a[i + 1][0];
        for (j = 0; j < N; j++)
                a[i][j] = a[i][j] * t;
}
```

1) Total number of memory accesses: [1]

2) Miss rate: [2]

3) Suppose you have the chance to double the cache size. Which following scheme is best choice for above code and why? (5')

Scheme A: double the number of sets. (i.e., 32 sets)
Scheme B: double the block size (i.e., 64 bytes)
Scheme C: double the number of ways (i.e., 8-way)