Practice Problem 4.13 (solution page 521)

Fill in the right-hand column of the following table to describe the processing of the `irmovq` instruction on line 4 of the object code in Figure 4.17:

Stage	Generic `irmovq V, rB`	Specific `irmovq $128, %rsp`
Fetch	icode:ifun ← M_1[PC] rA:rB ← M_1[PC + 1] valC ← M_8[PC + 2] valP ← PC + 10	
Decode		
Execute	valE ← 0 + valC	
Memory		
Write back	R[rB] ← valE	
PC update	PC ← valP	

How does this instruction execution modify the registers and the PC?

```
1   0x000: 30f20900000000000000 |     irmovq $9,  %rdx
2   0x00a: 30f31500000000000000 |     irmovq $21, %rbx
3   0x014: 6123                 |     subq %rdx, %rbx          # subtract
4   0x016: 30f48000000000000000 |     irmovq $128,%rsp         # Problem 4.13
5   0x020: 4043640000000000000  |     rmmovq %rsp, 100(%rbx)   # store
6   0x02a: a02f                 |     pushq %rdx               # push
7   0x02c: b00f                 |     popq  %rax               # Problem 4.14
8   0x02e: 734000000000000000   |     je done                  # Not taken
9   0x037: 804100000000000000   |     call proc                # Problem 4.18
10  0x040:                      | done:
11  0x040: 00                   |     halt
12  0x041:                      | proc:
13  0x041: 90                   |     ret                      # Return
14                              |
```

Figure 4.17 **Sample Y86-64 instruction sequence.** We will trace the processing of these instructions through the different stages.

Practice Problem 4.14 (solution page 522)

Fill in the right-hand column of the following table to describe the processing of the popq instruction on line 7 of the object code in Figure 4.17.

Stage	Generic popq rA	Specific popq %rax
Fetch	icode:ifun ← M_1[PC] rA:rB ← M_1[PC + 1] valP ← PC + 2	

Stage	Generic popq rA	Specific popq %rax
Decode	valA ← R[%rsp] valB ← R[%rsp]	
Execute	valE ← valB + 8	
Memory	valM ← M_8[valA]	
Write back	R[%rsp] ← valE R[rA] ← valM	
PC update	PC ← valP	

What effect does this instruction execution have on the registers and the PC?

Practice Problem 4.43 (solution page 530)

Suppose we use a branch prediction strategy that achieves a success rate of 65%, such as backward taken, forward not taken (BTFNT), as described in Section 4.5.4. What would be the impact on CPI, assuming all of the other frequencies are not affected?

Practice Problem 6.12 (solution page 699)

The problems that follow will help reinforce your understanding of how caches work. Assume the following:

- The memory is byte addressable.
- Memory accesses are to 1-byte words (not to 4-byte words).
- Addresses are 13 bits wide.
- The cache is two-way set associative ($E = 2$), with a 4-byte block size ($B = 4$) and eight sets ($S = 8$).

The contents of the cache are as follows, with all numbers given in hexadecimal notation.

2-way set associative cache

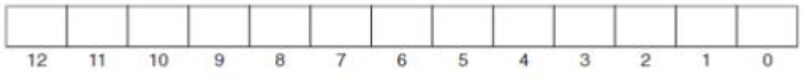
	Line 0						Line 1					
Set index	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3
0	09	1	86	30	3F	10	00	0	—	—	—	—
1	45	1	60	4F	E0	23	38	1	00	BC	0B	37
2	EB	0	—	—	—	—	0B	0	—	—	—	—
3	06	0	—	—	—	—	32	1	12	08	7B	AD
4	C7	1	06	78	07	C5	05	1	40	67	C2	3B
5	71	1	0B	DE	18	4B	6E	0	—	—	—	—
6	91	1	A0	B7	26	2D	F0	0	—	—	—	—
7	46	0	—	—	—	—	DE	1	12	C0	88	37

The following figure shows the format of an address (1 bit per box). Indicate (by labeling the diagram) the fields that would be used to determine the following:

CO. The cache block offset

CI. The cache set index

CT. The cache tag



Practice Problem 6.17 (solution page 701)

Transposing the rows and columns of a matrix is an important problem in signal processing and scientific computing applications. It is also interesting from a locality point of view because its reference pattern is both row-wise and column-wise. For example, consider the following transpose routine:

```
1   typedef int array[2][2];
2
3   void transpose1(array dst, array src)
4   {
5       int i, j;
6
7       for (i = 0; i < 2; i++) {
8           for (j = 0; j < 2; j++) {
9               dst[j][i] = src[i][j];
10          }
11      }
12  }
```

Assume this code runs on a machine with the following properties:

- `sizeof(int)` = 4.
- The `src` array starts at address 0 and the `dst` array starts at address 16 (decimal).
- There is a single L1 data cache that is direct-mapped, write-through, and write-allocate, with a block size of 8 bytes.
- The cache has a total size of 16 data bytes and the cache is initially empty.
- Accesses to the `src` and `dst` arrays are the only sources of read and write misses, respectively.

A. For each `row` and `col`, indicate whether the access to `src[row][col]` and `dst[row][col]` is a hit (h) or a miss (m). For example, reading `src[0][0]` is a miss and writing `dst[0][0]` is also a miss.

	dst array			src array	
	Col. 0	Col. 1		Col. 0	Col. 1
Row 0	m	_____	Row0	m	_____
Row 1	_____	_____	Row 1	_____	_____

B. Repeat the problem for a cache with 32 data bytes.

Practice Problem 6.18 (solution page 702)

The heart of the recent hit game *SimAquarium* is a tight loop that calculates the average position of 512 algae. You are evaluating its cache performance on a machine with a 2,048-byte direct-mapped data cache with 32-byte blocks ($B = 32$). You are given the following definitions:

```
1   struct algae_position {
2       int x;
3       int y;
4   };
5
6   struct algae_position grid[32][32];
7   int total_x = 0, total_y = 0;
8   int i, j;
```

You should also assume the following:

- `sizeof(int)` = 4.
- `grid` begins at memory address 0.
- The cache is initially empty.
- The only memory accesses are to the entries of the array `grid`. Variables `i`, `j`, `total_x`, and `total_y` are stored in registers.

Determine the cache performance for the following code:

```
1       for (i = 31; i >= 0; i--) {
2           for (j = 31; j >= 0; j--) {
3               total_x += grid[i][j].x;
4           }
5       }
6
7       for (i = 31; i >= 0; i--) {
8           for (j = 31; j >= 0; j--) {
9               total_y += grid[i][j].y;
10          }
11      }
```

A. What is the total number of reads?

B. What is the total number of reads that miss in the cache?

C. What is the miss rate?