Computer architecture handout assignment

Due date : 2014.12.15.

Student ID:
Name:

1. Permute the loops in the following function so that it scans the three-dimensional array A with a stride-1 reference pattern.

```
int sumarray3d(int A[N][N][N])
{
        int i, j, k, sum = 0;

        for (i = 0; i < N; i++) {
                for (j = 0; j < N; j++) {
                        for (k = 0; k < N; k++) {
                                sum += A[k][i][j];
                        }
                }
        }
        return sum;
}
```

2. In general, if the high-order *s* bits of an address are used as the set index, contiguous chunks of memory blocks are mapped to the same cache set.

A. How many blocks are in each of these contiguous array chunks?

B. Consider the following code that runs on a system with a cache of the form (S, E, B, m) = (512, 1, 32, 32):

```
int array[4096];

for (i = 0; i < 4096; i++)
        sum += array[i];
```

What is the maximum number of array blocks that are stored in the cache at any point in time?

3. 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)

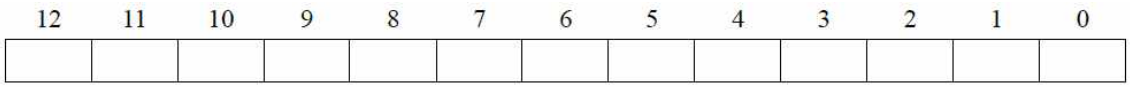
The contents of the cache are as follows, with all numbers given in hexadecimal notation.

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	1	DE	AD	BE	EF
6	91	1	A0	B7	26	2D	F0	0	-	-	-	-
7	46	0	-	-	-	-	DE	1	12	C0	88	37

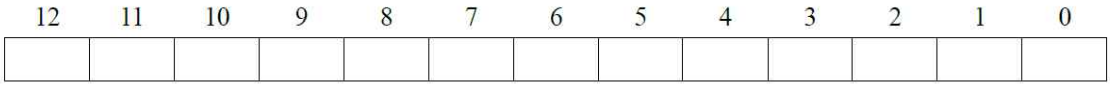
The following figure shows the format of an address (one bit per box). Indicate (by labeling the diagram) the fields that would be used to determine the following:

- CO The cache block offset
- CI The cache set index
- CT The cache tag



4. Suppose a program running on the machine in **Question 3** references the 1-byte word at address *0x0E36*. Indicate the cache entry accessed and the cache byte value returned in hex. Indicate the cache entry accessed and the cache byte value returned in hex. Indicate whether a cache miss occurs. If there is a cache miss, enter "-" for "Cache byte returned."

A. Address format (one bit per box):

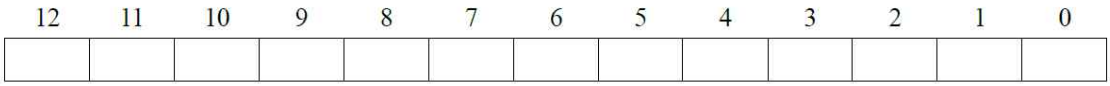


B. Memory reference:

Parameter	Value
Cache block offset (CO)	0x
Cache set index (CI)	0x
Cache tag (CT)	0x
Cache hit? (Y/N)	
Cache byte returned	0x

5. Repeat **Question 4** for memory address *0x0DD5*.

A. Address format (one bit per box):



B. Memory reference:

Parameter	Value
Cache block offset (CO)	0x
Cache set index (CI)	0x
Cache tag (CT)	0x
Cache hit? (Y/N)	
Cache byte returned	0x

6. The heart of the recent hit game SimAquarium is a tight loop that calculates the average position of 256 algae. You are evaluating its cache performance on a machine with a 1024-byte direct-mapped data cache with 16-byte blocks (B = 16). You are given the following definitions:

```
struct algae_position {
        int x;
        int y;
};

struct algae_position grid[16][16];
int total_x = 0, total_y = 0;
int i, j;
```

You should also assume:

- 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:

```
for (i = 0; i < 16; i++) {
        for (j = 0; j < 16; j++) {
                total_x += grid[i][j].x;
        }
}

for (i = 0; i < 16; i++) {
        for (j = 0; j < 16; j++) {
                total_y += grid[i][j].y;
        }
}
```

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? ______.

7. Explain how the memory accesses from your program are processed in a typical computer system. Your explanation should include concepts such as :

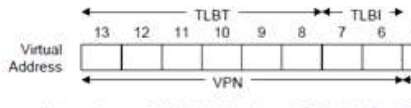
virtual address, physical address, TLB, TLB miss exception, exception handler in the operating system, page table, page fault, disk I/O, context switch (process switch), interrupt, interrupt handler in the operating system, process state (wait state / busy state), process scheduling by the operating system, first-level cache, second-level cache, and main memory(DRAM)

8~9.

A memory system assumption:

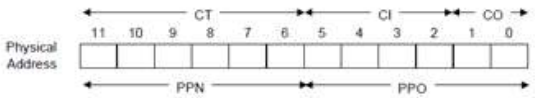
- The memory is byte addressable.
- Memory accesses are to **1-byte words** (not 4-byte words).
- Virtual addresses are 14 bits wide (n = 14).
- Physical addresses are 12 bits wide (m = 12).
- The page size is 64 bytes (P = 64).
- The TLB is four-way set associative with 16 total entries.
- VPN: the 2 low-order bits as set index (TLBI),
  the 6 high-order bits as the tag (TLBT).
- The L1 d-cache is physically-addressed and direct mapped, with a 4-byte line size and 16 total sets.
- Cache: the low-order 2 bits as the block offset (CO),
  the next 4 bits as the set index (CI),
  The remaining 6 bits as the tag (CT).

A snapshot of our memory system, including the TLB (a), a portion of the page table (b), and the L1 cache (c):



Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	34	1	02	–	0

(a) TLB: Four sets, sixteen entries, four-way set associative.



VPN	PPN	Valid
00	28	1
01	–	0
02	33	1
03	02	1
04	–	0
05	16	1
06	–	0
07	–	0

VPN	PPN	Valid
08	13	1
09	17	1
0A	09	1
0B	–	0
0C	–	0
0D	2D	1
0E	11	1
0F	0D	1

(b) Page table: Only the first sixteen PTEs are shown.

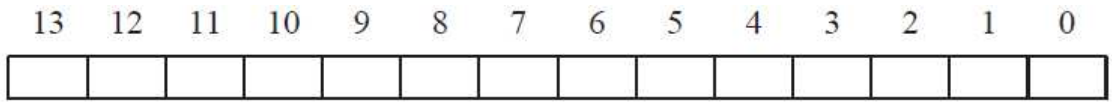
Idx	Tag	Valid	Blk 0	Blk 1	Blk 2	Blk 3
0	19	1	99	11	23	11
1	15	0	–	–	–	–
2	1B	1	00	02	04	08
3	36	0	–	–	–	–
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	31	0	–	–	–	–
7	16	1	11	C2	DF	03
8	24	1	3A	00	51	89
9	2D	0	–	–	–	–
A	2D	1	93	15	DA	3B
B	0B	0	–	–	–	–
C	12	0	–	–	–	–
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	14	0	–	–	–	–

(c) Cache: 16 sets, four-byte blocks, direct mapped.

8. Show how the memory system translates a virtual address into a physical address and accesses the cache. For the given address, indicate the TLB entry accessed, physical address, and cache byte value returned. Indicate whether the TLB misses, whether a page fault occurs, and whether a cache miss occurs. If there is a cache miss, enter "-" for "Cache byte returned". If there is a page fault, enter "-" for "PPN" and leave parts C and D blank.

Virtual address : *0x03d7*

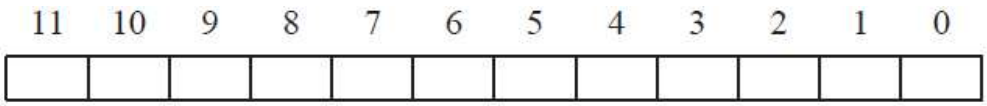
A. Virtual address format



B. Address translation

Parameter	Value
VPN	
TLB index	
TLB tag	
TLB hit? (Y/N)	
Page fault? (Y/N)	
PPN	

C. Physical address format



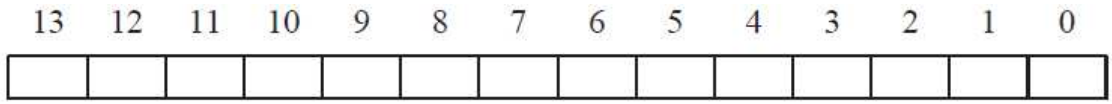
D. Physical memory reference

Parameter	Value
Byte offset	
Cache index	
Cache tag	
Cache hit? (Y/N)	
Cache byte returned	

9. Repeat **Question 8** for the following address:

Virtual address : *0x0040*

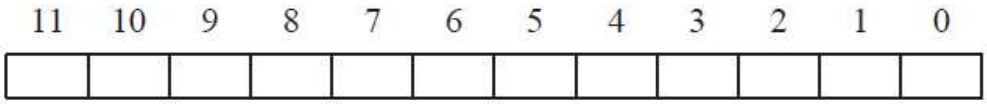
A. Virtual address format



B. Address translation

Parameter	Value
VPN	
TLB index	
TLB tag	
TLB hit? (Y/N)	
Page fault? (Y/N)	
PPN	

C. Physical address format



D. Physical memory reference

Parameter	Value
Byte offset	
Cache index	
Cache tag	
Cache hit? (Y/N)	
Cache byte returned	