CS 33: Computer Organization

Dis 1B: Week 10 Discussion

Agenda

- Final Review?

  or

- Forget about the finals and watch NCAA tournament?

Final

- March 20th, Monday: 3PM ~ 6PM
- Kinsey Pavillion 1220B

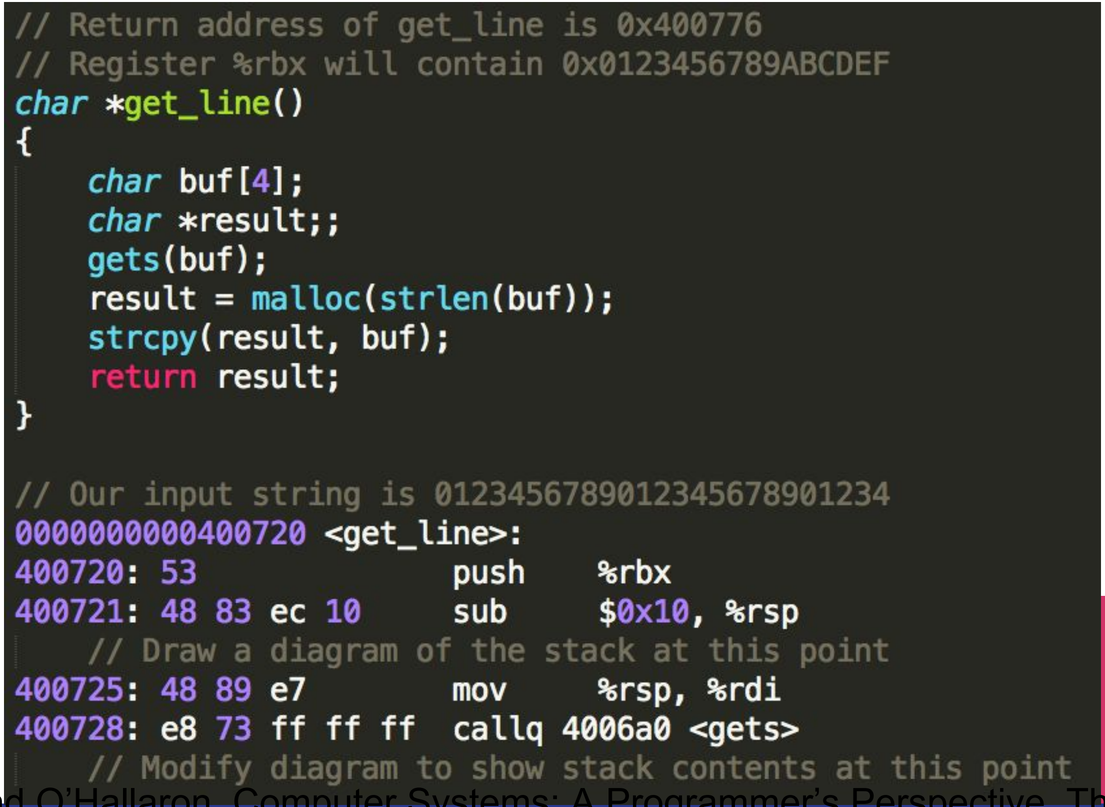
Things we covered after the midterm

- Buffer overflow
- Optimization
- Cache
- Processes
- Threading
- Multithreading
- Linking
- Virtual Memory

Final

- Most of the questions will be from the materials after the midterm
- The format of the final will be very similar to the midterm

Practice Problem 1



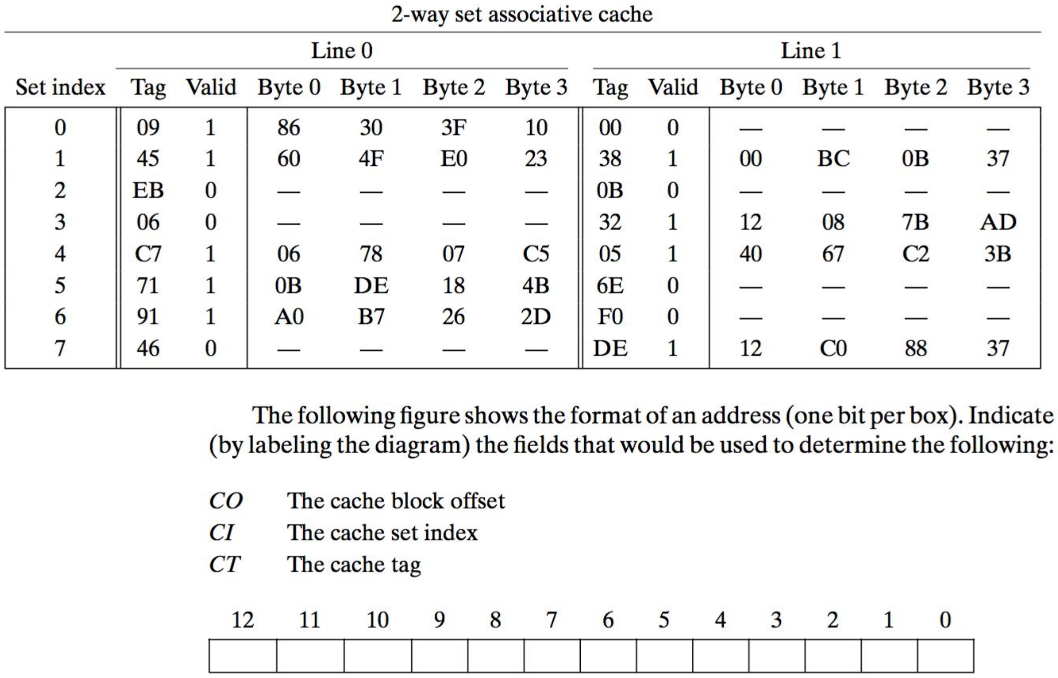
```
// Return address of get_line is 0x400776
// Register %rbx will contain 0x0123456789ABCDEF
char *get_line()
{
    char buf[4];
    char *result;;
    gets(buf);
    result = malloc(strlen(buf));
    strcpy(result, buf);
    return result;
}

// Our input string is 0123456789012345678901234
0000000000400720 <get_line>:
400720: 53              push    %rbx
400721: 48 83 ec 10     sub     $0x10, %rsp
    // Draw a diagram of the stack at this point
400725: 48 89 e7        mov     %rsp, %rdi
400728: e8 73 ff ff ff  callq 4006a0 <gets>
    // Modify diagram to show stack contents at this point
```

Adapted from Bryant and O’Hallaron, Computer Systems: A Programmer’s Perspective, Third Edition

Practice Problem 2

- Assume the following
  - The memory is byte addressable
  - Memory accesses are to 1-byte words
  - Addresses are 13 bits wide
  - The cache is two-way set associative, with 4-byte block size and 8 sets



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

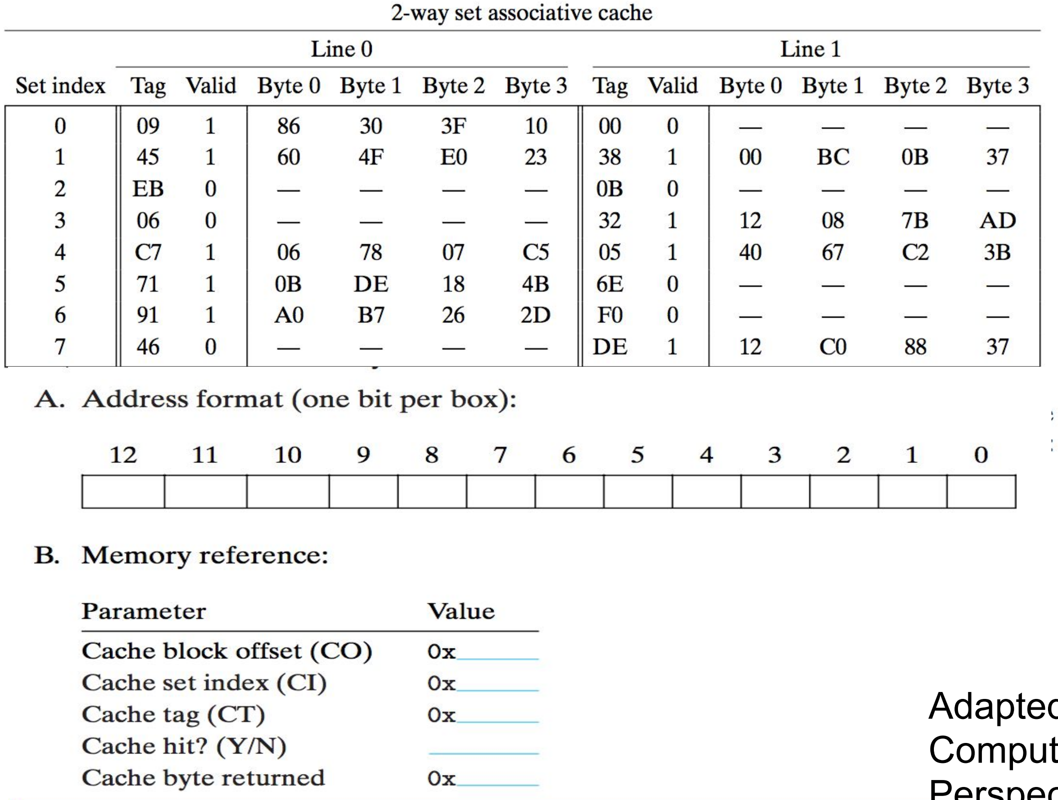
The following figure shows the format of an address (one bit per box). Indicate (by labeling the diagram) the fields that would be used to determine the following:

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

12	11	10	9	8	7	6	5	4	3	2	1	0

Adapted from Bryant and O'Hallaron, Computer Systems: A Programmer's Perspective, Third Edition

Practice Problem 2



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

A. Address format (one bit per box):

12	11	10	9	8	7	6	5	4	3	2	1	0

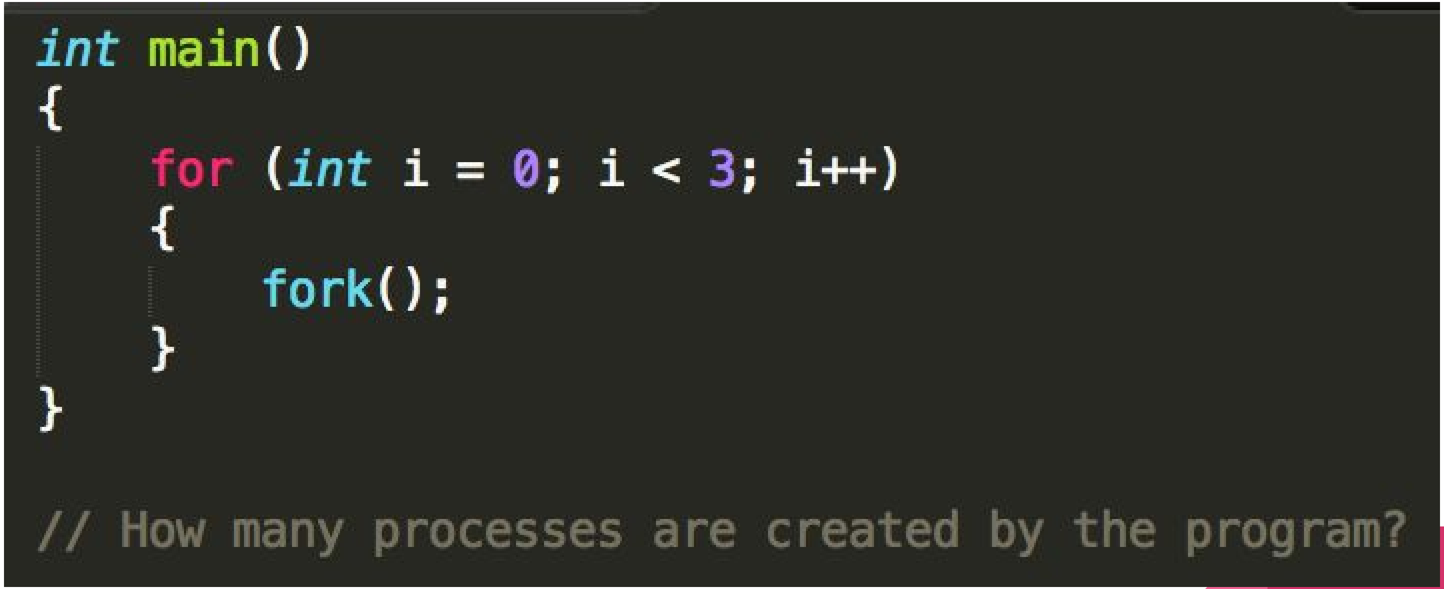
B. Memory reference:

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

- 0x0E34
- 0x0DD5
- 0x1FE4

Adapted from Bryant and O’Hallaron, Computer Systems: A Programmer’s Perspective, Third Edition

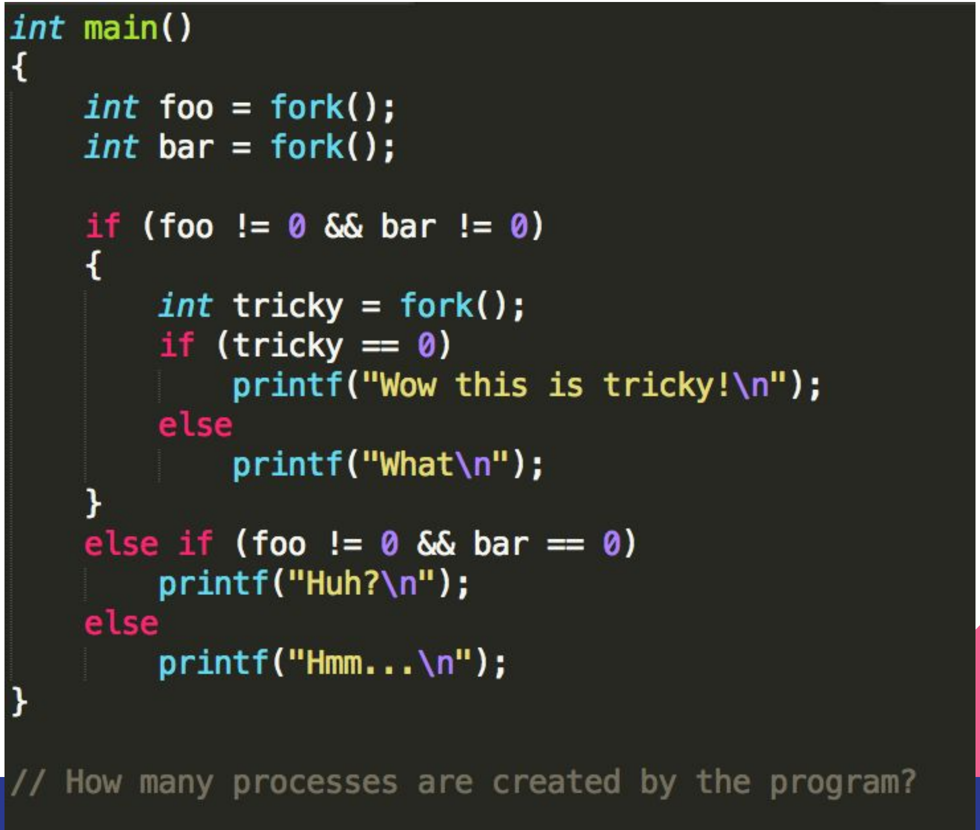
Practice Problem 3-1



```
int main()
{
    for (int i = 0; i < 3; i++)
    {
        fork();
    }
}

// How many processes are created by the program?
```

Practice Problem 3-2



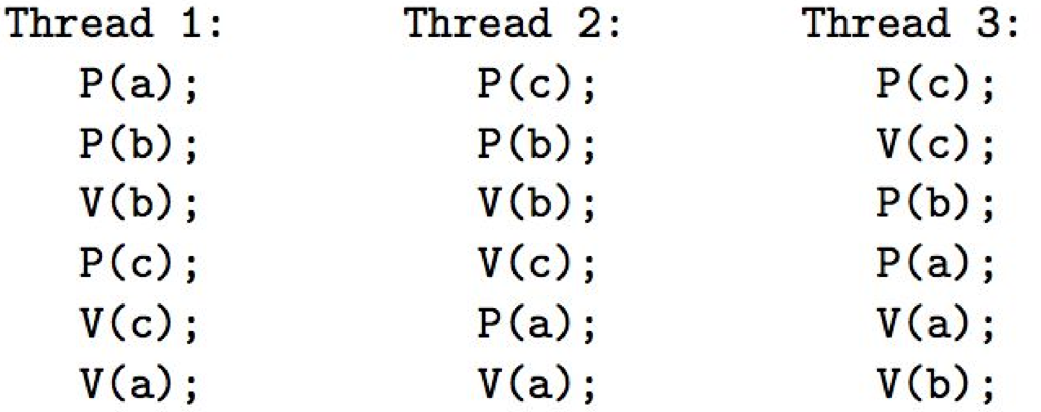
```c
int main()
{
    int foo = fork();
    int bar = fork();

    if (foo != 0 && bar != 0)
    {
        int tricky = fork();
        if (tricky == 0)
            printf("Wow this is tricky!\n");
        else
            printf("What\n");
    }
    else if (foo != 0 && bar == 0)
        printf("Huh?\n");
    else
        printf("Hmm...\n");
}

// How many processes are created by the program?
```

Practice Problem 4

- P locks the variable passed in as an argument
- V unlocks the variable passed in as an argument
- Will this program deadlock?

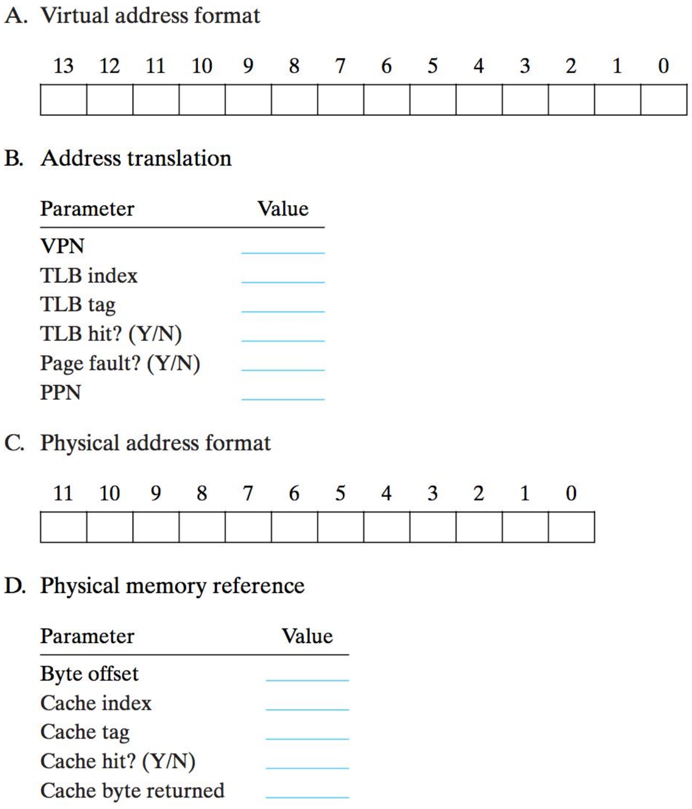


```
Thread 1:          Thread 2:          Thread 3:
    P(a);              P(c);              P(c);
    P(b);              P(b);              V(c);
    V(b);              V(b);              P(b);
    P(c);              V(c);              P(a);
    V(c);              P(a);              V(a);
    V(a);              V(a);              V(b);
```

Adapted from Bryant and O’Hallaron, Computer Systems: A Programmer’s Perspective, Third Edition

Practice Problem 5

- 0x03d4
- 0x03d7
- 0x03a9
- 0x0040



A. Virtual address format

13	12	11	10	9	8	7	6	5	4	3	2	1	0

B. Address translation

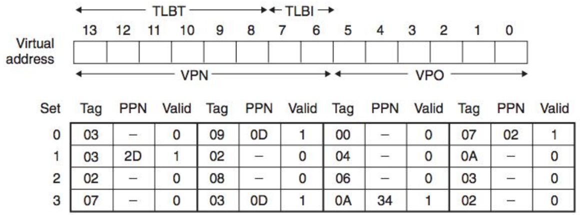
Parameter	Value
VPN	
TLB index	
TLB tag	
TLB hit? (Y/N)	
Page fault? (Y/N)	
PPN	

C. Physical address format

11	10	9	8	7	6	5	4	3	2	1	0

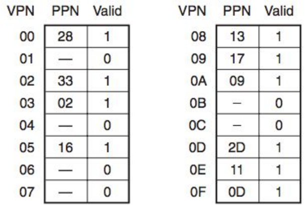
D. Physical memory reference

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


TLBT
TLBI
13 12 11 10 9 8 7 6 5 4 3 2 1 0
Virtual address
VPN
VPO

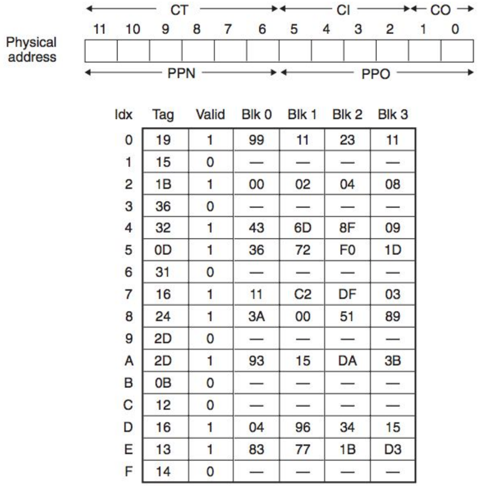
Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	34	1	02	–	0

(a) TLB: Four sets, 16 entries, four-way set associative



VPN	PPN	Valid	VPN	PPN	Valid
00	28	1	08	13	1
01	—	0	09	17	1
02	33	1	0A	09	1
03	02	1	0B	–	0
04	—	0	0C	–	0
05	16	1	0D	2D	1
06	—	0	0E	11	1
07	—	0	0F	0D	1

(b) Page table: Only the first 16 PTEs are shown


CT
CI
CO
11 10 9 8 7 6 5 4 3 2 1 0
Physical address
PPN
PPO

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: Sixteen sets, 4-byte blocks, direct mapped

Adapted from Bryant and O’Hallaron, Computer Systems: A Programmer’s Perspective, Third Edition

Good luck on your finals!

Thanks!