CS 213, Fall 1998
Homework Assignment H4
Assigned: Oct. 23, Due: Fri., Oct. 30, 11:59PM

Dave O'Hallaron (droh@cs.cmu.edu) is the lead person for this assignment.

The purpose of this assignment is to teach you how memory systems perform address translation.

Logistics

As usual, you may work in a group of up to 2 people. The only "hand-in" will be electronic. Use the following hardcopy sheets as worksheets. Any clarifications and revisions to the assignment will be posted on the Web page `assigns.html` in the class WWW directory.

All files for this assignment are in the directory:

```
/afs/cs.cmu.edu/academic/class/15213-f98/H4
```

The procedure is similar to H3. First create a (protected) directory to work in, and copy our template code using the command `tar -xvf /afs/cs.cmu.edu/academic/class/15213-f98/H4/H4.tar`. This will create the files `Makefile`, `checker.c`, and `handin.h`. In this assignment you will modify `handin.h` and hand it in with the command `make handin NAME=yourname` where `yourname` is replaced with the Andrew Id of the first person on your team. Your solution will be copied to the handin directory. If you need to submit another version to fix a bug, use the command

```
make handin NAME=yourname VERSION=versionnumber
```

where `versionnumber` starts at 2 and counts up with each submission *after* the initial handin.

To check that you've formatted your answers correctly, run the command `make checker` and then run `./checker`, the resulting program. The output of this program shows you the way we'll interpret your answers when grading (but doesn't tell you whether or not your answer is correct). Please check your solution and fix any mistakes before turning it in.

Problem 1

In each line of the table below, you are given a list of cache parameters: n bits of physical address, and an E-way associative cache with a block size of B bytes and a size of D data bytes. For each list of parameters, compute the corresponding number of cache sets (S), the total number of bits (C) needed to implement the cache, and the number of tag bits (t), set index bits (s), and block offset bits (b). For your computation of C, assume that for each block there are an additional t tag bits and 1 valid bit, and that for each set there are an additional $E(E-1)$ bits that implement the LRU replacement policy.

For this problem, type all numbers in decimal form, with no exponents or abbreviations. For example, write "1024" instead of "1k".

	n	D	B	E	S	t	s	b	C
1.	32	1024	4	1					
2.	32	1024	4	4					
3.	32	1024	4	256					
4.	32	1024	8	1					
5.	32	1024	8	4					
6.	32	1024	8	128					
7.	32	1024	32	1					
8.	32	1024	32	4					
9.	32	1024	32	32					

Virtual Address Translation

The following problems concerns the components of the memory hierarchy and the way virtual addresses are translated into physical addresses.

- The memory is byte addressable.
- Virtual addresses are 14 bits wide.
- Physical addresses are 12 bits wide.
- The page size is 64 bytes.
- The TLB is 4-way set associative with 16 total entries.
- The cache is direct mapped, with a 4 byte line size and 16 total lines.

In the following tables, all numbers are given in hexadecimal. The contents of the TLB are as follows.

TLB												
Set Index	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	03	21	0	09	0D	1	00	13	0	07	02	1
1	03	2D	1	02	13	0	04	21	0	0A	11	0
2	02	00	0	08	1F	0	06	2F	0	03	11	0
3	07	29	0	03	0D	1	0A	34	1	02	1B	0

The page table for the first 16 pages and the cache are as follows:

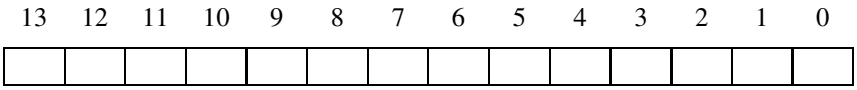
Page Table					
VPN	PPN	Valid	VPN	PPN	Valid
000	28	1	008	13	1
001	2B	0	009	17	1
002	33	1	00A	09	0
003	02	1	00B	1A	0
004	2A	0	00C	27	0
005	16	1	00D	2D	1
006	04	0	00E	11	1
007	26	0	00F	0D	1

Cache						
Index	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3
0	19	1	99	11	23	11
1	15	0	55	59	0B	41
2	1B	1	00	02	04	08
3	36	0	84	06	B2	9C
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	31	0	A2	37	68	10
7	16	1	11	C2	DF	03
8	24	1	3A	00	51	89
9	2D	0	46	92	88	FE
A	11	1	93	15	DA	3B
B	0B	0	71	49	10	05
C	12	0	82	03	02	12
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	14	0	20	07	12	19

Problem 2

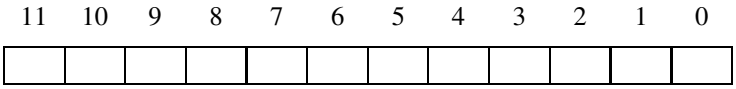
A. The box below shows the format of a virtual address. Indicate (by labeling the diagram) the fields that would be used to determine the following:

VPO	The virtual page offset
VPN	The virtual page number
TLBI	The TLB index
TLBT	The TLB tag



B. The box below shows the format of a physical address. Indicate (by labeling the diagram) the fields that would be used to determine the following:

PPO	The physical page offset
PPN	The physical page number
CO	The byte offset within the cache line
CI	The cache index
CT	The cache tag



For the following problems, you are to show how the memory system translates a virtual address into a physical address and accesses the cache.

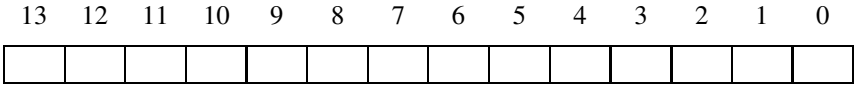
For the given virtual address, indicate the TLB entry accessed, the physical address, and the cache byte value returned. Indicate whether the TLB misses, whether a page fault occurs, and whether a cache miss occurs.

Assume that each access (load instruction) is for a single byte (unlike the Alpha, which accesses 8-byte words).

Problem 3

Virtual address: `027C`

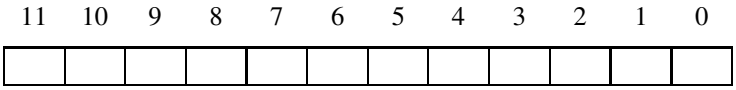
A. Virtual address format



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

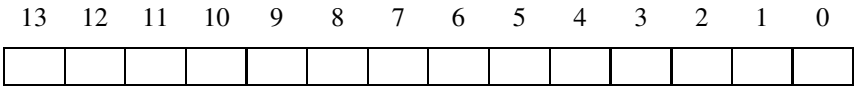
Parameter	Value
Byte offset	
Cache Index	
Cache Tag	
Cache Hit? (Y/N)	
Cache Byte returned	

Note: if there is a cache miss, enter “0x0” for “Cache Byte returned”.

Problem 4

Virtual address: `03D4`

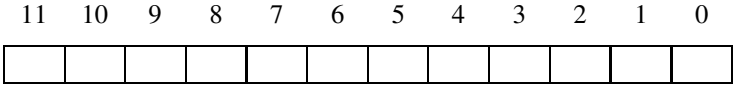
A. Virtual address format



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

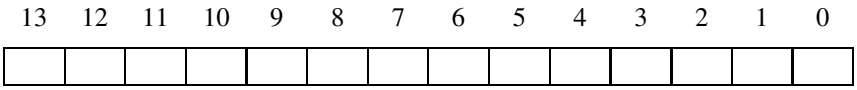
Parameter	Value
Byte offset	
Cache Index	
Cache Tag	
Cache Hit? (Y/N)	
Cache Byte returned	

Note: if there is a cache miss, enter “0x0” for “Cache Byte returned”.

Problem 5

Virtual address: `03A9`

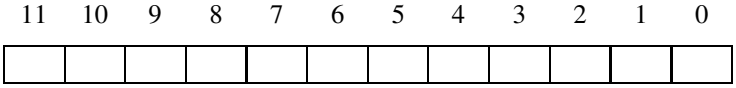
A. Virtual address format



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	

Note: if there is a cache miss, enter “0x0” for “Cache Byte returned”.