Name:	UID:

For Discussion Credit, it's required to fill out this survey: tinyurl.com/LA-Feedback-F25

1. Calculate the Size

(a) Given a 32-bit virtual address space and a 24-bit physical address, determine the number of bits in the VPN, VPO PPN, and PPO for the following page sizes:

Page Size	VPN bits	VPO bits	PPN bits	PPO bits
1 KiB				
2 KiB				
4 KiB				
8 KiB				

1 KiB (Kibibyte) = 1024 Bytes; 1 KB (Kilobyte)= 1000 Bytes

(b) For each of the following caches and a 16-bit physical address, fill in the table with the appropriate number of bits (assume that we are doing 1-byte cache accesses)

Cache	Tag bits	Index bits	Offset bits	Total Data in Cache
Direct-mapped cache with 32 sets and a cache line size of 4 bytes				
2-way set associative cache with 128 sets and a cache line size of 32 bytes				
8-way set associative cache with 64 sets and a cache line size of 1 byte				
16-way fully associative cache and a cache line size of 64 bytes				

2. What's that Data?

You are given the following details about the memory system and the states of the TLB and caches.

- 16-bit physical address, 12-bit virtual address, 16-byte page size (2^4 bytes)
- 2-way set associative TLB with 16 entries
- 8-way set associative cache with 2 sets and a cache block size of 4 bytes
- The replacement policy for the TLB and cache is LRU, with most recently used on top

TLB			
Index	Tag	Valid	PPN
0	15	1	A80
0	01	1	408
1	0D	1	D6C
1	00	0	D8D
2	08	1	4C2
2	1F	0	5BE
3	0D	0	490
3	02	1	9C2
4	14	1	FA3
4	11	1	CA4
5	05	1	5AD
5	10	1	C9D
6	19	1	0CE
6	0F	1	E7C
7	10	1	087
7	01	0	D87

Page Table					
VPN	Valid	PPN	VPN	Valid	PPN
01	0	D8D	68	1	490
05	0	8F6	78	1	087
0D	1	A21	7B	0	823
0F	0	9FD	7F	0	99D
10	0	0EB	82	1	4F1
13	1	9C2	88	1	CA4
16	1	A80	8F	1	408
2F	0	CE8	94	0	821
30	0	899	A4	1	FA3
31	1	9D1	D0	1	80C
34	1	0CE	D4	1	D87
4E	0	B1B	D6	1	C9D
58	0	DDE	EC	0	BBC
5F	1	5AD	ED	1	49C
63	0	8C7	F2	1	187
69	1	D6C	F8	0	769

Cache						
Index	TAG	Valid	Data			
0	0B5A	1	85	7F	ED	E2
0	036F	1	6E	41	B4	ED
0	1442	1	99	25	DA	18
0	1F46	0	11	92	A1	78
0	019D	0	10	48	16	F9
0	1B0F	1	86	08	16	FD
0	0137	0	46	CF	FA	F7
0	0353	0	5D	46	F3	9E
1	010F	1	63	2A	BD	80
1	0939	1	F6	C3	ED	A9
1	0920	0	BC	54	DF	04
1	193B	1	B9	31	F2	3F
1	1AD9	1	29	80	4C	77
1	13A3	1	D4	52	B0	7A
1	1385	1	FE	F0	3D	89
1	118C	0	FB	40	47	C8

Fill in the following each of the following tables.

Virtual Address	**0x31E**
VPN	
VPO/PPO	
TLB Index	
TLB Tag	
TLB Hit? (Y/N)	
Page Fault? (Y/N)	
PPN	
Physical Address	
Cache Offset	
Cache Index	
Cache Tag	
Data (1-byte access)	

Virtual Address	**0x0D3**
VPN	
VPO/PPO	
TLB Index	
TLB Tag	
TLB Hit? (Y/N)	
Page Fault? (Y/N)	
PPN	
Physical Address	
Cache Offset	
Cache Index	
Cache Tag	
Data (1-byte access)	

Virtual Address	**0x0F4**
VPN	
VPO/PPO	
TLB Index	
TLB Tag	
TLB Hit? (Y/N)	
Page Fault? (Y/N)	
PPN	
Physical Address	
Cache Offset	
Cache Index	
Cache Tag	
Data (1-byte access)	

Virtual Address	**0xA40**
VPN	
VPO/PPO	
TLB Index	
TLB Tag	
TLB Hit? (Y/N)	
Page Fault? (Y/N)	
PPN	
Physical Address	
Cache Offset	
Cache Index	
Cache Tag	
Data (1-byte access)	

3. Who wins the Race?

For the following code, draw the process graph and provide a possible output.

```
void Tortoise_and_Hare() {
    int child_status;
    printf("The Race Begins!\n");
    if (fork() == 0) {
        printf("The hare speeds through the race!\n");
        // The hare enters the barn to take a nap
        if (fork() == 0) {
            printf("The farmer comes back home!\n");
            exit(0)
        } else {
            printf("zzzz...\n");
            wait(&child_status);
            printf("The hare is awoken and continues the race!\n");
        }
        fork(); // the hare races to the finish with their friends
    } else {
        printf("The Tortoise crawls through the race!\n");
    }
    printf("Finish!\n");
}
```