TLB			
Index	Tag	PPN	Valid
0	09	4	1
	12	2	1
	10	0	1
	08	5	1
	05	7	1
	13	1	0
	10	3	0
	18	3	0
1	04	1	0
	0C	1	0
	12	0	0
	08	1	0
	06	7	0
	03	1	0
	07	5	0
	02	2	0

Page Table					
VPN	PPN	Valid	VPN	PPN	Valid
00	6	1	10	0	1
01	5	0	11	5	0
02	3	1	12	2	1
03	4	1	13	4	0
04	2	0	14	6	0
05	7	1	15	2	0
06	1	0	16	4	0
07	3	0	17	6	0
08	5	1	18	1	1
09	4	0	19	2	0
0A	3	0	1A	5	0
0B	2	0	1B	7	0
0C	5	0	1C	6	0
0D	6	0	1D	2	0
0E	1	1	1E	3	0
0F	0	0	1F	1	0

2-way Set Associative Cache												
Index	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3
0	19	1	99	11	23	11	00	0	99	11	23	11
1	15	0	4F	22	EC	11	2F	1	55	59	0B	41
2	1B	1	00	02	04	08	0B	1	01	03	05	07
3	06	0	84	06	B2	9C	12	0	84	06	B2	9C
4	07	0	43	6D	8F	09	05	0	43	6D	8F	09
5	0D	1	36	32	00	78	1E	1	A1	B2	C4	DE
6	11	0	A2	37	68	31	00	1	BB	77	33	00
7	16	1	11	C2	11	33	1E	1	00	C0	0F	00

Details

Memory is byte addressable

Physical addresses: 13 bits; Virtual addresses: 16 bits

Page size: 512 bytes

Cache: 4 byte lines, 16 lines, 2-way associative

TLB: 16 entries, 8-way associative

Questions:

Show bits for PPN, PPO, VPN, VPO, CT, CI, CO

Look up virtual address 1DDE in page table, cache

(Showed look up VPN in TLB)

Recall Slide #71

```
#include <sys/mman.h>

void *mmap(void *addr, size_t length,
           int prot, int flags,
           int fd, off_t offset);

int munmap(void *addr, size_t length);
```

mmap changes the page table:

- **addr** — address to map or **NULL** for kernel choice
- **length** – bytes to map rounded up to page size
- **prot** — bitwise **PROT_{READ,WRITE,EXEC}**
- **flags** — **MAP_{PRIVATE,SHARED}**, maybe **MAP_ANON**
- **fd** — file to map into memory if not **MAP_ANON**
- **offset** — offset into file

"Unsafe" code.

Want to use different untrusted processes, and give them each only half the secret.

How to use mmap to make this safe?

```c
void compute_initial_secrets(int *secret1, int *secret2);
void adjust_secrets(int *secret1, int *secret2, int i);

void check1(int *secret1, int *answer1, int i);
void check2(int *secret2, int *answer2, int i);

static int *alloc() {
  return malloc(SIZE * sizeof(int));
}

int main()
{
  int *secret1 = alloc();
  int *secret2 = alloc();
  int *answer1 = alloc();
  int *answer2 = alloc();
  int i;

  compute_initial_secrets(secret1, secret2);

  for (i = 0; i < ITERS; i++) {
    adjust_secrets(secret1, secret2, i);

    check1(secret1, answer1, i);
    check2(secret2, answer2, i);

    if (memcmp(answer1, answer2, SIZE * sizeof(int))) {
      printf("disagree %d\n", i);
      exit(1);
    }
  }

  printf("ok\n");
  return 0;
}
```