Solutions to Sample Problems for CS2400 Final

December 10, 2014

1

The following problem concerns the way virtual addresses are translated into physical addresses.

- The memory is byte addressable.

- Memory accesses are to **1-byte words** (not 4-byte words).

- The TLB is 2-way set associative with 8 total entries.

- The L1 Cache is direct mapped, with a 4-byte block size and 64 total bytes.

- Virtual addresses are 12 bits wide.

- Physical addresses are 10 bits wide.

- The page size is 64 bytes.

In the following tables, **all numbers are given in hexadecimal**. The contents of the TLB and a portion of the page tables are as follows:

Page Table		
VPN	PPN	Present
034	00c	1
030	000	1
032	009	1
001	004	1
03f	00d	1
02f	00a	1
00e	006	1
02c	003	1
021	00f	1
012	00b	1
03c	008	1
03d	002	1
006	001	1
024	00e	1
017	005	1
003	007	1

TLB			
Index	Tag	PPN	Valid
0	d	0c	1
	-	–	0
1	0	04	0
	-	–	0
2	4	0b	1
	1	01	0
3	f	0d	1
	-	–	0

Cache			
Index	Valid	Tag	Data
0	1	C	6021130E
1	1	0	DCAEB820
2	0	2	1DFE0C46
3	0	B	29E5DBF8
4	1	9	DFFBCC85
5	1	2	CB570940
6	1	8	57A84A44
7	1	C	8E85761F
8	1	B	DF2C1CE2
9	1	7	BE10CEA4
10	1	0	579C4AB6
11	1	C	A11D81A1
12	1	3	B250AE92
13	1	5	7751E21A
14	0	C	6AA3E19A
15	1	F	6AC09E41

1. **[4 Points]** The box below shows the format of a virtual address. Indicate (by labeling the diagram) the fields (if they exist) that would be used to determine the following: (If a field doesn't exist, don't draw it on the diagram.)

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

11	11	11	11	11	11	5	4	3	2	1	0
VPN	VPN	VPN	VPN	VPN	VPN	VPO	VPO	VPO	VPO	VPO	VPO

  TLBT   TLBT   TLBT   TLBT   TLBI   TLBI

2. **[2 Points]** 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) and *PPN* (The physical page number).

9	8	7	6	5	4	3	2	1	0
PPN	PPN	PPN	PPN	PPO	PPO	PPO	PPO	PPO	PPO

3. **[24 Points]** (12 points each) For the given virtual addresses, indicate the TLB entry accessed and the physical address. **Indicate whether the TLB misses and whether the entry is or is not in the page table.** If the physical page number and address can not be determined, write "N/A". Then if a physical address exists indicate the cache translation parts, if its a cache hit, and a value if applicable. If any part can't be determined just write "N/A".

Virtual address: `0x4a3`

  (i) Virtual address: (one bit per box)

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

(iii) Physical address: (one bit per box)

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

(ii) Address translation

Parameter	Value
VPN	0x12
TLB Index	0x2
TLB Tag	0x4
TLB Hit? (Y/N)	Y
In Page Table? (Y/N)	Y
PPN	0xB

(iv) Cache Translation

Parameter	Value
PPN	0xB
Block Offset	0x3
Cache Index	0x8
Cache Tag	0xB
Cache Hit? (Y/N)	Y
Byte Value	0xE2

Virtual address: `0x38d`

(i) Virtual address: (one bit per box)

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

(iii) Physical address: (one bit per box)

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

(ii) Address translation

Parameter	Value
VPN	0xE
TLB Index	0x2
TLB Tag	0x3
TLB Hit? (Y/N)	N
In Page Table? (Y/N)	Y
PPN	0x6

(iv) Cache Translation

Parameter	Value
PPN	0x6
Block Offset	0x1
Cache Index	0x3
Cache Tag	0x6
Cache Hit? (Y/N)	N
Byte Value	N/A

2

1. 4.

2. Any of 2, 3, or 4.

3

```
C Code:

myFunction(int x, char a ) {
    signed int b[5];
    unsigned short i;
    signed int y = 8;

    for (i=x;i>0;i--) {
        if (y>0) {
            b[i-2] = a + i;
        }
        else {
            b[i+5] = a;
        }
        y = y + 1 + b[i];
    }
    return y;
}
```

4

Answer these questions on linking

1. **[15 Points]** For the following code, identify the symbols listed in the symbol table of the ELF relocatable object files (.o), whether that symbol is defined or undefined, and if defined, then in which section of the corresponding ELF file that the symbol would be defined.

main.c

```
void  swap();

int  x=7;

int  main(){
    int  z;
    swap(x);
    z = x;
    return  0;
}
```

swap.c

```
int  y=4;
int  *bufp1 = &y;

void  swap(int  x){
    int  *bufp0;

    *bufp0 = x;
    temp = *bufp0;
    *bufp0 = *bufp1;
    *bufp1 = temp;
}
```

main.o

Symbol Name	Defined/undefined	Section
swap	undefined	–
x	defined	.data
main	defined	.text

swap.o

Symbol Name	Defined/undefined	Section
temp	defined	.bss
y	defined	.data
bufp1	defined	.data
swap	defined	.text