Full Name: _____

"On my honor as a University of Colorado at Boulder student I have neither given nor received unauthorized assistance on this work."

CSCI 2400, Fall 2014

Sample Problems for Final Exam

Instructions:

- Make sure that your exam is not missing any sheets, then write your full name on the front.
- Write your answers in the space provided below the problem. Show your work. If you make a mess, clearly indicate your final answer.
- Feel free to use the back of pages, but indicate that you have done so.
- This exam is CLOSED BOOK and you can use a *single page* of notes along with our reference sheets. You can not use a computer or calculator.

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:

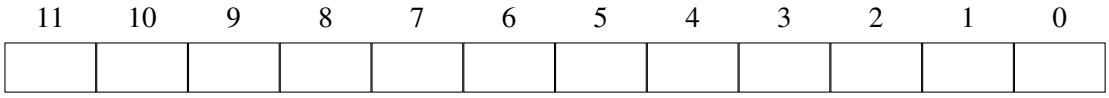
TLB			
Index	Tag	PPN	Valid
0	d	0c	1
	-	–	0
1	0	04	0
	-	–	0
2	4	0b	1
	1	01	0
3	f	0d	1
	-	–	0

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

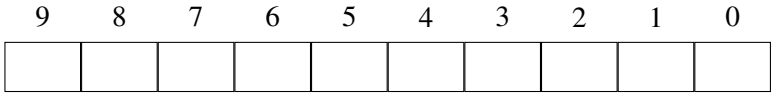
Cache			
Index	Valid	Tag	Data
0	1	C	6021130E
1	1	0	DCAEB820
2	0	2	1DFE0C46
3	0	B	29E5DBF8
4	1	9	DFFBCC85
5	1	2	CB570940
6	1	8	57A84A44
7	1	C	8E85761F
8	1	B	DF2C1CE2
9	1	7	BE10CEA4
10	1	0	579C4AB6
11	1	C	A11D81A1
12	1	3	B250AE92
13	1	5	7751E21A
14	0	C	6AA3E19A
15	1	F	6AC09E41

(a) **[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



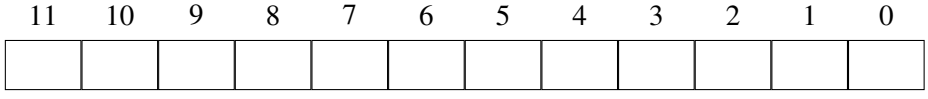
(b) **[2 Points]** The box below shows the format of a physical address. Indicate (by labeling the diagram) the fields that would be used to determine the following: *PPO* (The physical page offset) and *PPN* (The physical page number).



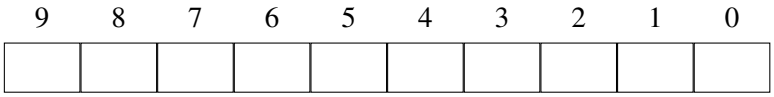
(c) **[24 Points]** (12 points each) For the given virtual addresses, indicate the TLB entry accessed and the physical address. **Indicate whether the TLB misses and whether the entry is or is not in the page table.** If the physical page number and address can not be determined, write "N/A". Then if 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)



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



(ii) Address translation

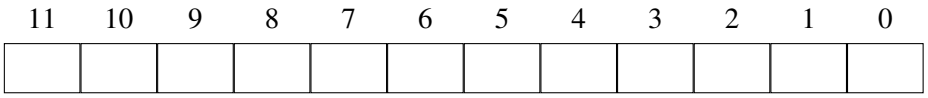
Parameter	Value
VPN	0x
TLB Index	0x
TLB Tag	0x
TLB Hit? (Y/N)	
In Page Table? (Y/N)	
PPN	0x

(iv) Cache Translation

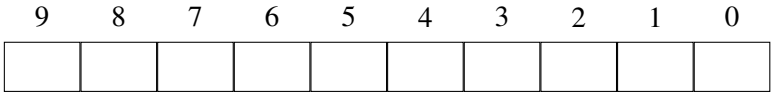
Parameter	Value
PPN	0x
Block Offset	0x
Cache Index	0x
Cache Tag	0x
Cache Hit? (Y/N)	
Byte Value	0x

Virtual address: `0x38d`

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



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



(ii) Address translation

Parameter	Value
VPN	0x
TLB Index	0x
TLB Tag	0x
TLB Hit? (Y/N)	
In Page Table? (Y/N)	
PPN	0x

(iv) Cache Translation

Parameter	Value
PPN	0x
Block Offset	0x
Cache Index	0x
Cache Tag	0x
Cache Hit? (Y/N)	
Byte Value	0x

2.
```
int count = 0;

void handler(int sig){
    count ++;
}

int  main(){
    signal(SIGCHLD, handler);

    for (int i = 0; i < 4; i++){
        if (!fork()){
            exit(0);
        }
    }
    while (wait(NULL) != -1); //wait for all children to die.
    printf("count = %d\n", counter);
}
```

(a) How many `SIGCHLD` signals get generated as this program executes?

(b) What is/are the possible value(s) for count?

3. Look at the assembly code below and and fill out the blanks in the corresponding C code.

C Code:

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

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

```
myFunction:

push   %ebp
mov    %esp,%ebp
mov    %edi,-0x34(%ebp)
mov    %esi,%eax
mov    %al,-0x38(%ebp)
movl   $0x8,-0x24(%ebp)
mov    -0x34(%ebp),%eax
mov    %ax,-0x26(%ebp)
jmp    L1

.L4
cmpl   $0x0,-0x24(%ebp)
jle    L2
movzwl -0x26(%ebp),%eax
lea    -0x2(%eax),%ecx
movsbl -0x38(%ebp),%edx
movzwl -0x26(%ebp),%eax
add    %eax,%edx
movslq %ecx,%eax
mov    %edx,-0x20(%ebp,%eax,4)
jmp    L3

.L2
movzwl -0x26(%ebp),%eax
add    $0x5,%eax
movsbl -0x38(%ebp),%edx
mov    %edx,-0x20(%ebp,%eax,4)

.L3
mov    -0x24(%ebp),%eax
lea    0x1(%eax),%edx
movzwl -0x26(%ebp),%eax
mov    -0x20(%ebp,%eax,4),%eax
add    %edx,%eax
mov    %eax,-0x24(%ebp)
movzwl -0x26(%ebp),%eax
sub    $0x1,%eax
mov    %ax,-0x26(%ebp)

.L1
cmpw   $0x0,-0x26(%ebp)
jne    L4
mov    -0x24(%ebp),%eax
pop    %ebp
ret
```

4. Answer these questions on linking

(a) **[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.o

Symbol Name	Defined/undefined	Section