

name \_\_\_\_\_

## ICS 331 Midterm Solutions

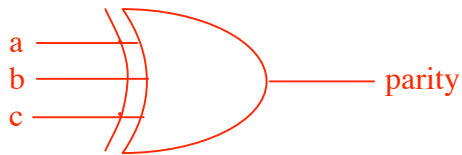
Open book, open notes, closed computer

### 1. Circuit Design (4 pts)

Design a circuit to generate an even parity bit for a 3-bit register with bits  $a$ ,  $b$ , and  $c$  (even parity means that the sum of the bits and the parity bit is always even). First draw the truth table for the even parity bit. Next give a minimal equation for the even parity bit. Finally design and draw a circuit to compute the parity bit using any combination of 1 input NOT gates, 2 or 3 input AND, OR, NAND, NOR, XOR and/or XNOR gates.

abc	parity
000	0
001	1
010	1
011	0
100	1
101	0
110	0
111	1

$$\text{parity} = \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} + abc$$



## 2. Caches and Virtual Memory (4 pts)

Assume: a 4-way set associative cache with block size 2, cache size 16K entries; memory page size of 4K; word/address size of 24 bits; bits numbered starting with the LSB = 0. Hint: it may be helpful to draw some pictures to help you envision the architecture.

- a. Which bits of an address word are used to determine the block number?  
bit 0
- b. Which bits of an address word are stored as the tag in the cache?  
bits 12 to 23 (highest 12 bits)
- c. Which bits of an address are used as an index into the cache?  
bits 1 to 11
- d. How many entries are stored in each set of the cache (one of the 4 sets)?  
4K
- e. How many bits are needed for the cache including only tags and values?  
each block is 12 bits of tag + 48 bits of value (block size of 2 \* 24 bits) = 60 bits/block  
60 bits/block \* 8K blocks = 480K bits
- f. In an address, which bits are used for the offset within a page?  
bits 0 to 11
- g. In an address, which bits are used for the page number?  
bits 12 to 23 (highest 12 bits)
- h. In an 8K entries TLB with presence, modified and two use bits, how many bits total are needed for the TLB?  
(each entry is 12 bits for the virtual page number, 12 bits for the real page number, 1 bit presence, 1 bit modified, and 2 bits use = 28 bits/entry) 28\*8K=224K bits

### 3. Assembly Language Programming (4 pts)

Translate this MASM x86 assembly language program into Sun Sparc assembly language:

*MASM x86 program:*

```
title Recursive Factorial

.model small
.stack

.data

.code
main proc
.startup
    push 5    ; compute 5!
    call factorial
.exit
main endp

factorial proc
    push bp ;save base ptr
    mov bp,sp ; new base ptr
    push bx ; save old bx
    mov bx,[bp+4] ; bx <- N
    cmp bx,0
    jnz recurse
    mov ax,1 ; f(0) = 1
    jmp done
recurse:
    mov ax,bx ; ax <- N
    dec ax ; ax <- N-1
    push ax ; f(N-1)
    call factorial
    mul bl ; ax <- result
done:
    pop bx ; restore bx
    pop bp ; restore base
    ret 2
factorial endp

end
```

*Sparc program:*

```
! input parameter belongs in %o0
! return result in %o0
!==== Minimal prologue ====
.section ".text"
.global main
.align 4
main: save %sp,-96,%sp !minimum!
!=====
! add your code here:
set 5, %o0 ! compute 5!
call factorial
nop
!====Standard epilogue ====
ret !Return to the OS.
restore !Delay slot - done first

factorial:
    save %sp, -96, %sp
    cmp %i0, 0
    bne recurse
    nop
    set 1, %o0
    ba done
    nop
recurse:
    mov %i0, %o0
    dec %o0
    call factorial
    nop
    umul %o0, %i0, %o0
done:
    ret
    restore
!=====
! The data section
.section ".data"
```

**4. Microcode (3 pts)**

Given the following horizontal microcode, design the microcode for this instruction:

MUL (R0+4), R1, (R2) ! multiple (R0+4) and R1, store in (R2)

(R0+4) is indirect addressing, meaning use register R0=0000 as a memory address, add 4 to it and use the contents of that memory location. R1=0001. (R2) is indirect addressing, meaning use register R2=0010 as a memory address and store the result there. All operations take one clock cycle.

<b>ALU</b>	000 No operation 001 Add 010 Subtract 011 Multiply 100 Divide 101 Left shift operand 1 110 Right shift operand 1 111 Increment operand 1
<b>operand 1 or 2</b>	0 No operation 1 Load value from data transfer mechanism
<b>result</b>	0 No operation 1 Send value to data transfer mechanism
<b>register interface</b>	00xxxx No operation 01xxxx Move register xxxx to data transfer mechanism 10xxxx Move data transfer mechanism to register xxxx 11xxxx Move constant xxxx to data transfer mechanism
<b>AGU</b>	00xxxx No operation 01xxxx Move memory (address in register xxxx) to data transfer mechanism 10xxxx Move data transfer mechanism to memory (address in register xxxx) 11xxxx No operation

Your microcode translation:

ALU			op1	op2	res	Register interface						AGU					
0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	1	0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0