Nick Lautner NLautner@purdue.edu

CS250 Homework Midterm Practice

a) Solve this homework and turnin your handwritten or printed answers on March 26th 2012 during the midterm exam. You will find similar questions in the midterm exam.
I will post the solutions on Sunday afternoon. I strongly suggest you to complete your homework on your own befor looking at the solutions.

b) If you submit your solutions nicely printed in HTML or PDF format and turn them in before Friday March 23rd 11:59pm you will get 1% extra over your total grade.
To submit your solutions copy your PDF or html files into a directory hw1-sol/ in lore and then type
turnin -c cs250 -p hw1-sol hw1-sol

==========

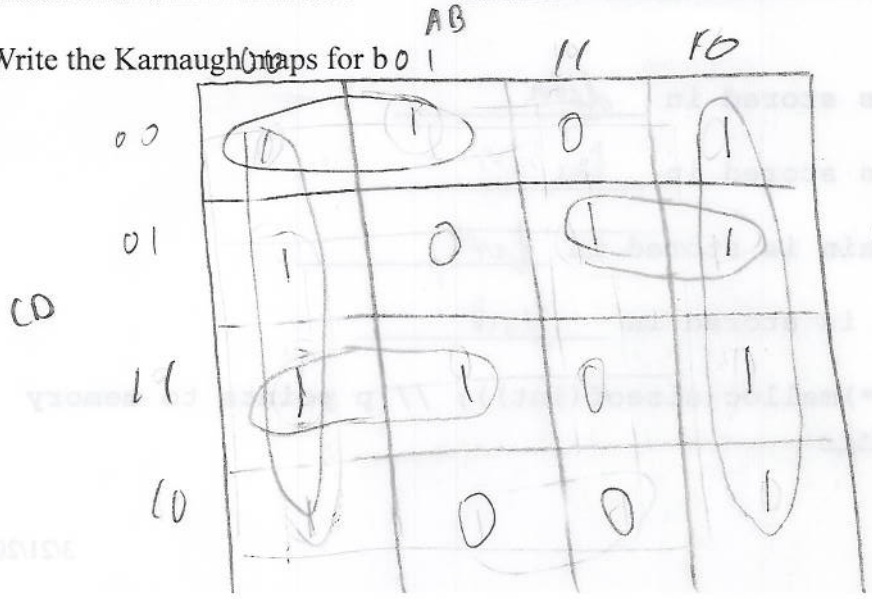
1. Implement the boolean function for segment b for a 7-segment decoder. Using four switches w, x, y, z choose the digits 0, 1, 2, 3,4,5,6,7,8,9,A,B,C,D,E,F in the seven segment display.

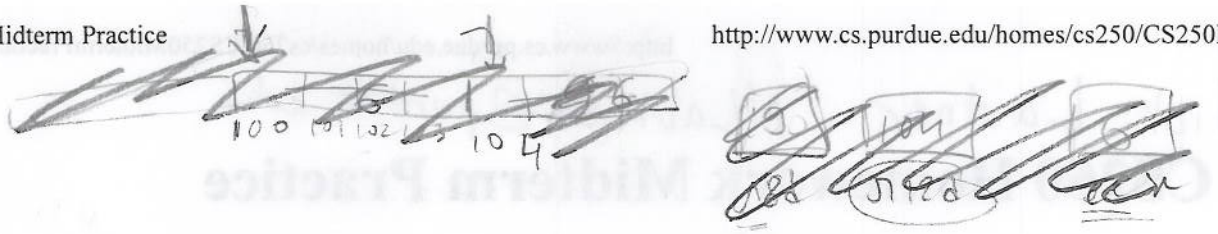
a) Write the truth table for the segment b

w	x	y	z	b	
0	0	0	0	1	0
0	0	0	1	1	1
0	0	1	0	1	2
0	0	1	1	1	3
0	1	0	0	1	4
0	1	0	1	0	5
0	1	1	0	0	6
0	1	1	1	1	7

w	x	y	z	b	
1	0	0	0	1	8
1	0	0	1	1	9
1	0	1	0	1	A
1	0	1	1	1	B
1	1	0	0	0	C
1	1	0	1	1	D
1	1	1	0	0	E
1	1	1	1	0	F

b) Write the Karnaugh maps for b



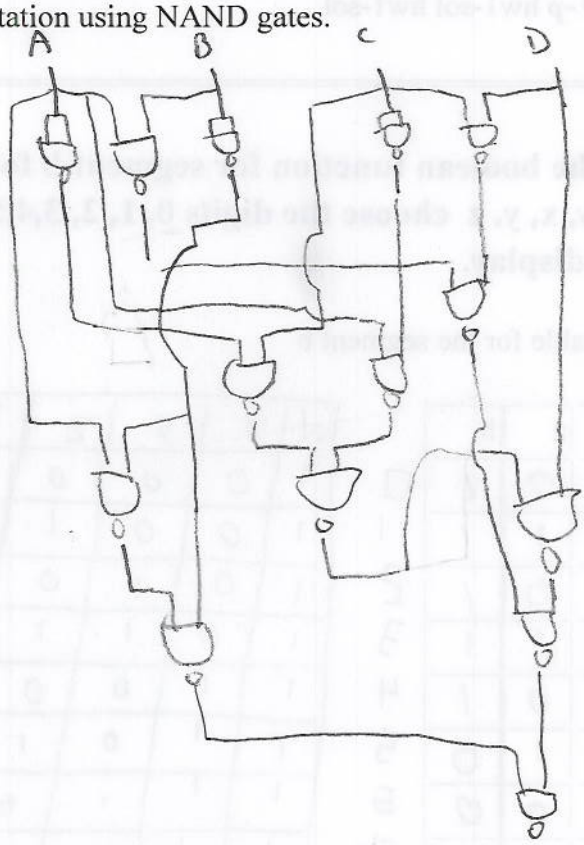


c) Using the Karnaugh maps, write the boolean expressions for segment b using a minimum "sum of products" boolean expression.

b = $A'B'D + A'B'C + A'CD + B + AC + AB'$

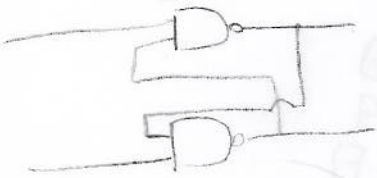
$A'B'(D+C) + D(A'C+AC') + B + AB'$

d) Draw the schematic of your implementation using NAND gates.



2. Draw a Flip Flop

a) Using NAND gates



b) Using NOR gates

3. In each line write the name of the memory section where the variables are stored:

```
int a = 5;  // a is stored in data

int b[20];  // b is stored in bss

int main() {  // main is stored in text
{
    int x;    // x is stored in Stack

    int *p = (int *)malloc(sizeof(int)); // p points to memory
stored in heap
```

}

4. Enumerate the steps needed to load a program.

allocate space for all sections of executable file
load executable and libraries into memory
link to libraries

5. What is a "Dynamic Linker" and a "Static Linker".

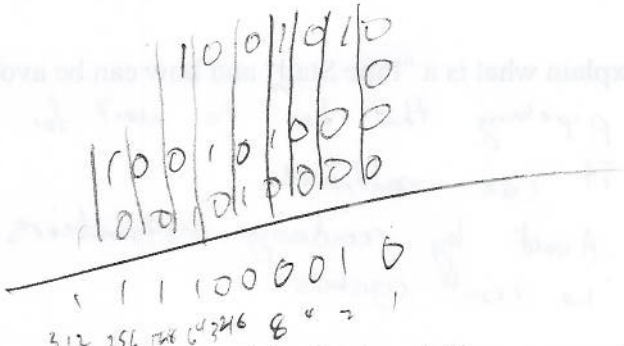
Linkers are used to add blocks of code for a program to use when it executes
Static means the code is added when it compiles and any changes to the linked libraries requires re-compiling
Dynamic means changes don't need a recompile

6. Perform the following multiplication in binary. Also, show convert to decimal the operands and the result.

74 13 962

1001010 x 1101

1111000010 = 962



7. Perform the following division in binary, Also, convert to decimal the operands and the result.

11011 = 27

1011 | 100101001 = 256 + 32 + 8 + 1 = 297

-1011
01111
-1011
01000
-1011
01011
-1011
0000

11011 = 27

8. Represent the number1.25 in binary using the IEEE 754 double representation. The formula is given:

Val in decimal = $(-1)^s$ x (1.m) x $2^{(e\text{-bias})}$ where:

bias = 1023
s = bit 63 = 0
e = bits 52 to 62
m= bits 0 to 51

Val in binary: 0 01111111111 01000

9. Explain what is a "Pipe Stall" and how can be avoided.

pipelining that has to wait for a previous instruction to finish before it can continue

Avoid by reordering instructions, avoid unnecessary branches, delay references to result registers

10. Obtain the representation of number -58 using 8 bits. Verify that 78 + (-58) is equal to 20 using binary addition and complements of 2.

11000110

11000110 = -58 0111010
01001110 = 78 1000101
00010100 = 20 1000110

11. Write a function "int isBigEndian()" that will return 1 if the computer is big endian or 0 if it is little endian.

```
unsigned int u = 0x12345678
unsigned char *p = (unsigned char *)&u
if (*p == 0x12) return 1
else return 0
```

12. What is the difference between Harvard Architecture and Von Newman Architecture

Harvard = program and data memory are in diff spaces

VN = ... are in the same space

13. What do CISC and RISC stand for and enumerate the characteristics of each.

CISC - Complex instruction set computer, instructions are not basic and may require more than 1 cycle

RISC - reduced instruction set computer, 1 cycle instructions, good for pipelining

14. Write the following characteristics of the ATMEGA328:

Program Memory (KB): 32
RAM (KB): 2
Clock Speed (Hz): 16 - 20
Number of I/O ports: 14 - 23
Number of A/D ports: 6

14. Rewrite the following program in AVR assembly language. Remember that int's in AVR are 2 bytes long.

```
int a = 6;
int b = 3;
int c;

void setup() {
  c = a + b;
}

loop()
{
}
```

```
.global setup
        loop
.type   setup, @function
.type   loop, @function
setup:
lds r24, b
lds r25, (b)+1
lds r18, a
lds r19, (a)+1
add r24, r18
adc r25, r19
sts (c)+1, r25
sts c, r24
ret
loop:
ret
```

```
.global a
        b
        c
.type   a, @object
        b, @object
        c, @object
.size   a, 2
        b, 2
        c, 2
a:      .word 6
b:      .word 3
c:
```

15. Rewrite the following program in AVR assembly language. Remember that int's in AVR are 2 bytes long.

```
int a = 6;
int b = 4;
int c;

void setup() {
 c = a * b;
}

loop()
{
}
```

```
.global a
        b
        c
.section .bss   .data a, "aw", @progbits
                .data b, "aw", @progbits
                .data c, "aw", @progbits
.type   a, @object
        b, @object
        c, @object
        setup, @function
        loop, @function
.size   a, 2
        b, 2
        c, 4
```

```
a:      .word 6
b:      .word 4
c:
setup = lds r20, b
        lds r21, (b)+1
        lds r18, a
        lds r19, (a)+1
        mul r20, r18
        movw r24, r0
        mul r20, r19
        add r25, r0
        mul r21, r18
        add r25, r0
```

clr r1
sts (c)+1, r25
sts c, r24
ret

16. In x86-64 Assembly language implement the function **int addarray(int n, int * array)** that add all the elements of the array
passed as parameter. n is the length of the array.

```
mov (%esi), %eax
mov $1, %edx
cmp %edx, %edi
je end
```

```
start: add $4, %esi
       add %esi, %eax
       inc %edx
       cmp %edi, %edx
       jle start
       jmp end
end: ret
```

17. In x86-64 Assembly language implement a program "maxmin" that prompts two integer numbers fom stdin and then displays the maximum, minimum and average of both: numbers. Here is an example of the usage:

```
bash> maxmin
a=5
b=8
max=8
min=5
avg=6
```

```
.data   sf:  .string "%d"
        pf:  .string "a=%d\n"
        pf1: .string "b=%d\n"
        pf2: .string "max=%d\n"
        pf3: .string "mn=%d\n"
        pf4: .string "avg=%d\n"
        .comm n 8
.globl  main
.type   main, @function
main:
        movq $sf, %rdi
        movq $n, %rsi
        scanf
        movq $n, %rdx
        scanf
        movq $n, %rcx
```

```
other:  movq $pf2, %rdi
        movq %rdx, %rsi
        printf
        movq $pf3, %rdi
        movq $rcx, %rsi
        printf
notLr:  movq $pf2, %rdi
        movq %rcx, %rsi
        printf
        movq $pf3, %rdi
        movq %rdx, %rsi
        printf
```

```
        movq $pf, %rdi
        movq $n, %rsi
        printf
        movq $pf1, %rdi
        movq %rdx, %rsi
        printf
        cmp (%rdx), (%rcx)
        jl other
        jge notLr
        movq $0, %rax
        movq %rdx, %rax
        addq %rcx, %rax
        divq $2, %rax
        ret
```