

159233 Tutorial 1

Section A

1. How many possible three input, one output binary logic functions are there?
 a) 256 b) 64 c) 8 d) 16

2. The following truth table is for which logic gate?

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

- a) AND b) OR c) NOT d) XOR
3. Which of the following is NOT true?
 a) An eight bit adder can be made from two four bit adders
 b) An adder is a sequential logic device
 c) A full adder has three inputs
 d) A four bit adder can be made from four full adders

4. What is the truth table for the borrow output from a 3 bit subtractor (performing x-y-z)

a)	x	y	z	b	b)	x	y	z	b	c)	x	y	z	b	d)	x	y	z	b
	0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0
	0	0	1	0		0	0	1	1		0	0	1	1		0	0	1	1
	0	1	0	0		0	1	0	1		0	1	0	1		0	1	0	1
	0	1	1	1		0	1	1	0		0	1	1	1		0	1	1	1
	1	0	0	0		1	0	0	1		1	0	0	0		1	0	0	0
	1	0	1	1		1	0	1	0		1	0	1	0		1	0	1	1
	1	1	0	1		1	1	0	0		1	1	0	0		1	1	0	1
	1	1	1	1		1	1	1	1		1	1	1	1		1	1	1	1

5. What is the name for a logic device that can choose one of a number of inputs as its output.
 a) Decoder b) Multiplexor c) Demultiplexor d) XOR gate

Section B

6. For the following truth table:

a	b	c	d
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

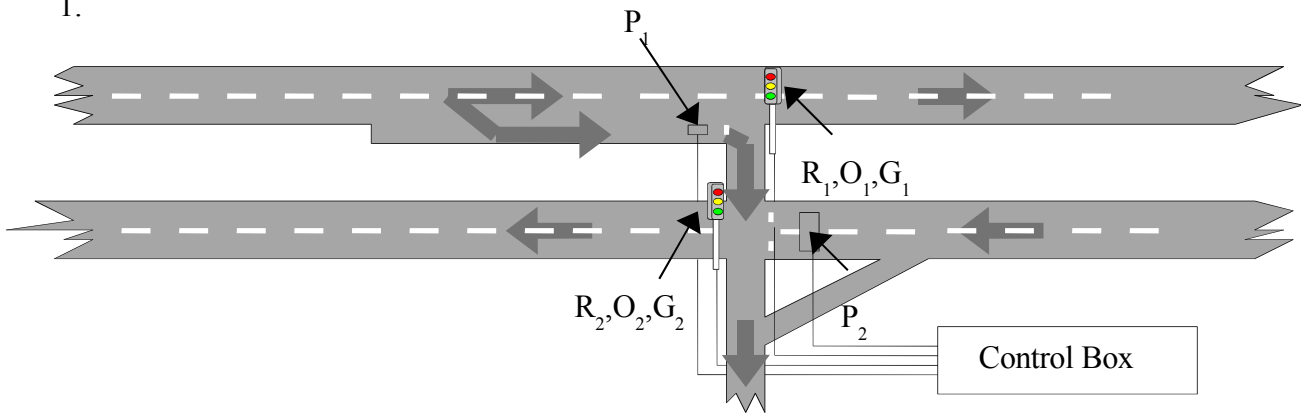
Give a boolean expression for d in terms of a,b and c. Draw a logic diagram for this function.

7. Redraw the logic diagram for question 6 using only NAND and NOT gates.
8. Design a voting circuit with four inputs and one output, the output should only be one if three or more of the inputs are one.

Draw the truth table, give a boolean expression and draw the logic diagram for this function.

Section B

1.



A simple road junction is shown above. The main road is a highway with two lanes on each side. The junction is with a one way road and traffic lights are used to allow vehicles to cross the oncoming traffic.

The control box has two inputs P_1 and P_2 that are connected to sensors under the road. These have a value of 1 if there is a vehicle present above the sensor and 0 otherwise. The box also has outputs which go to two traffic lights (light 1 and light 2). Each traffic light contains three lamps, red, orange and green, thus the outputs are composed of six signals R_1, O_1, G_1 and R_2, O_2, G_2 . The control box has a clock with a cycle time of 5 seconds.

If there are no vehicles present on P_1 then R_1 and G_2 will be on. When a vehicle is detected by P_1 then light 2 will turn orange. After five seconds, light 1 will turn green and light 2 will turn red. The lights will stay in this state while there is a vehicle on P_1 and there is no vehicle on P_2 . When this is no longer the case, then light 1 will turn orange. After five seconds, light 1 will turn red, light 2 will turn green and traffic will flow along the highway as before.

- Draw an ASM chart for your controller.
- Give Boolean expressions for the states.
- Draw your multiplexor driven controller

2.

A child's toy is being designed to help increase co-ordination skills. It has three buttons (red_button, green_button and blue_button), and three lights (red_light, green_light and blue_light).

The toy's controller (which has a cycle time of 1 second) obeys these rules:

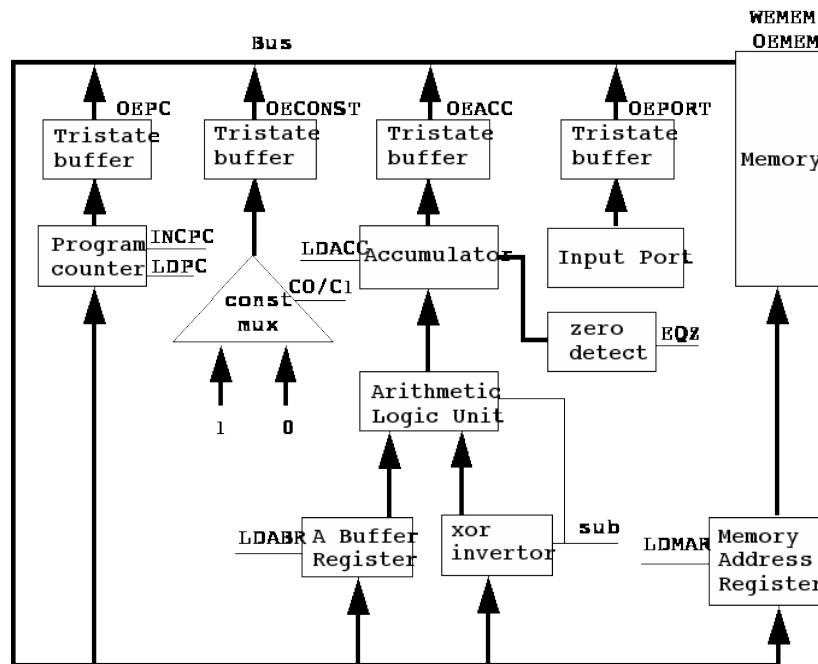
- Wait until the child presses one or more of the buttons.
- If the child presses one button, the corresponding coloured light is lit for 1 second.
- If the child presses two buttons together, the light with the same colour as the button that has not been pressed is lit for 1 second.
- If the child presses all three buttons together, all the lights are lit for 1 second.
- Go back to waiting.

For the controller of the above toy:

- Draw an ASM chart
- Give Boolean expressions for the states.
- Design a circuit, including the output signals

159233 Tutorial 4

The following is a block diagram of the picocomputer datapath



Section A

1. What is the purpose of the MAR register?
 - a) It holds address of the next instruction.
 - b) It holds data to be written to memory
 - c) It holds an address for a memory operation
 - d) It holds the current instruction
2. Why is the ABR register necessary?
 - a) To enable addition and subtraction of two numbers.
 - b) To hold the result of an addition or subtraction.
 - c) To pass an address to the Accumulator
 - d) To invert one of the inputs to the ALU for subtraction.
3. The Picocomputer has an 8 bit data bus and an 8 bit address bus, what would be the approximate maximum size of a typical program (assuming no data storage is necessary)?
 - a) 50 instructions
 - b) 80 instructions
 - c) 140 instructions
 - d) 200 instructions
4. What is the sequence of control signals necessary for the ADD instruction (assume opcode has been read already).
 - a) OECONST,LDABR→OEPC,LDMAR,OEACC→OEMEM,LDMAR→OEMEM,LDACC
 - b) OEACC,LDABR→OEPC,LDMAR,INCPC→OEMEM,LDMAR→OEMEM,LDACC
 - c) OECONST,LDABR→OEPC,LDACC,INCPC→OEPC,LDMAR→OEMEM,LDACC
 - d) OEACC,LDABR→OEMAR,LDACC,INCPC→OEPC,LDMAR→OEMEM,LDACC
5. What is the sequence of control signals necessary to add one to the accumulator.
 - a) OECONST,LDABR→OEPC,LDMAR→OEMEM,LDMAR→OEMEM,LDACC
 - b) OECONST,C0/C1,LDABR→OEMEM,LDMAR→OEMEM,LDACC
 - c) OECONST,C0/C1,LDABR→OEPC,LDACC→OEPC,LDMAR→OEMEM,LDACC
 - d) OECONST,C0/C1,LDABR→OEACC,LDACC

Section B

1. Describe how you would implement the following new pico-computer instruction.
LDA @address - Load Acc with the value stored in memory at the address given by the value stored in memory at the address specified by the operand (memory indirect addressing).
2. What type of device could be used to construct the Program Counter for the pico-computer?
3. Describe how you would implement the following new pico-computer instruction.
SWAP address - swap the value in the Accumulator with the value in memory at the specified address
4. Programs for the pico-computer will often use self modifying code, briefly explain why? Suggest an extra instruction that could be used to avoid this problem.

159233 Tutorial 5

Section A

1. If the gate delay through a full adder is $2\mu\text{s}$ and the delay through all other gates is $1\mu\text{s}$, what is the delay through a 4 bit ripple carry adder?
a) $4\mu\text{s}$ b) $5\mu\text{s}$ c) $8\mu\text{s}$ d) $12\mu\text{s}$
2. What is the delay for question 1 if a 4 bit carry lookahead adder is used?
a) $4\mu\text{s}$ b) $5\mu\text{s}$ c) $8\mu\text{s}$ d) $12\mu\text{s}$
3. A CPU has 300 control signals and a control unit with a microprogram that contains 3000 instructions, what size is the microprogram counter (μPC)?
a) 8 bits b) 10 bits c) 12 bits d) 14 bits
4. For the CPU in question 3, roughly what size is the microprogram ROM? Assume it uses horizontal microcode.
a) 942,000 bits c) 886,000 bits
b) 756,000 bits d) 1024,000 bits
5. A Microprogrammed control unit will generally be slower than a hardwired control unit, why?
a) Because each instruction will take more steps to execute.
b) Because the gate delay through the μPC is longer than through the flip flops.
c) Because the microprogram will be longer than the number of states in the ASM diagram.
d) Because the gate delay through the microcode ROM and address selection logic is longer than through the multiplexors.
6. What is the next address selection logic in the microcode control unit used for?
a) Generating the next value for the μPC
b) Generating the next value for the PC
c) Reading an instruction from the instruction register.
d) Holding the address of the current μ instruction in the μ code ROM.

Section B

1. Show the steps involved in multiplying 4×-3 using Booth's algorithm.
2. Draw a diagram of a circuit (without controller) that could be used to multiply two numbers using Booth's algorithm. Use a single adder with an XOR inverter to perform subtraction.
3. Draw an ASM chart for the controller in question 2.
4. Briefly explain how you think the microcode patch mechanism in a modern CPU may work.

159233 Tutorial 6

1. What is an important difference between a five address machine and a three address machine?
 - a) The three address machine has a PC, the five address machine does not.
 - b) For the three address machine, one source is the same as the destination.
 - c) The five address machine has separate jump instructions.
 - d) The three address machine has only one type of instruction.
2. Programs for a one address machine will:
 - a) Contain more instructions than programs for a two address machine.
 - b) Contain fewer instructions than programs for a two address machine.
 - c) Use more memory than programs for a two address machine.
 - d) Use the same memory as programs for a two address machine.
3. What is the following instruction:
ADD r1,[r2+r3]
 - a) Zero address
 - b) One address
 - c) Two address
 - d) Three address
4. What type of machine is the Pico-computer:
 - a) Zero address
 - b) One address
 - c) Two address
 - d) Three address
5. What addressing mode does the second operand in the instruction for question 3 use.
 - a) Register
 - b) Register Indirect
 - c) Displacement
 - d) Indexed
6. Which of the following addressing modes does the picocomputer have.
 - a) Indexed
 - b) Direct
 - c) Memory Indirect
 - d) Immediate
7. A CPU has instructions using 2 addresses and an ALU operation, the operation is performed on data stored at the two addresses and the result stored at the first address. If a number 'a' is stored at address 19, what does the following do?
XOR 20,20
ADD 20,19
ADD 19,19
ADD 19,20
 - a) Multiply a by 2
 - b) Multiply a by 3
 - c) Multiply a by 4
 - d) Multiply a by 5
8. Which of the following CPU's will probably run the same program faster
 - a) A 3200 Mhz RISC CPU that executes an instruction in 2 clock cycles
 - b) A 1600 Mhz CISC CPU that executes an instruction in 1 clock cycle
 - c) A 2000 Mhz CISC CPU that executes an instruction in 2 clock cycles
 - d) A 2000 Mhz RISC CPU that executes an instruction in 2 clock cycles

159233 Tutorial 7

1. Write a complete program in 2051 assembly language to find the sum of the first 11 numbers in the Fibonacci sequence (0,1,1,2,3,5,8,13,21,34,55,89). Use registers to hold the current number in the sequence, the previous number and the sum so far. The result should be stored in a memory location labelled "sum".
2. Write a program, in 2051 assembler, that will alter an array of characters by changing any non-numerical character into 00h. (The numerical values 0 - 9 are \equiv 30h - 39h)

eg Array before running your program: 23h 31h B5h 34h 35h 78h 31h3Fh
 Array after running your program: 00h 31h 00h 34h 35h 00h 31h00h

The array of numbers is located at addresses 40h to 4Fh inclusive.

3. Write a program, in 2051 assembler, that will convert an array of numbers into their ASCII equivalents.

The array of numbers is located at addresses 40h to 4Fh inclusive. The number values are restricted to 00h - 0Fh so that each can be replaced by a single ASCII character '0' - 'F', ie the ASCII equivalents of the numbers in the array.

eg if location 40h contains 06h it will be replaced by 36h which is equivalent to '6', if it contains 0Bh it will be replaced by 42h which is equivalent to 'B'.

(The equivalent values are: 0 - 9 \equiv 30h - 39h, A - F \equiv 41h - 46h)

Tutorial 7 Solutions

1. Write a complete program in 2051 assembly language to find the sum of the first 11 numbers in the Fibonacci sequence (0,1,1,2,3,5,8,13,21,34,55,89). Use registers to hold the current number in the sequence, the previous number and the sum so far. The result should be stored in a memory location labelled "sum".

```
; in c
; int sum,prevoius,current,next;
; sum=1;
; previous=0;
; current=1;
;   for(int i=10;i>0;i--) {
;       next=current+previous;
;   previous=current;
;   current=next;
;   sum=sum+current;
; }
```

```
$MOD2051
    sum data 08h
    previous equ r0
    current equ r1
    current_data 1
    i equ r2
cseg
    clr a
    mov previous,#0
    mov current,#1
    mov sum,current
    mov i,#10
loop: mov a,current ; a is next
      add a,previous
      mov previous,current_
      mov current,a
      add a,sum
      mov sum,a
      djnz i,loop
end
```

2. Write a program, in 2051 assembler, that will alter an array of characters by changing any non-numerical character into 00h. (The numerical values 0 - 9 are \equiv 30h - 39h)

eg Array before running your program: 23h 31h B5h 34h 35h 78h 31h3Fh
 Array after running your program: 00h 31h 00h 34h 35h 00h 31h00h

The array of numbers is located at addresses 40h to 4Fh inclusive.

```
$MOD2051

    s equ r0
    mov s,#40h
loop1:mov a,@s
      clr c
      subb a,#30h
      jc zero
      subb a,#10
      jc nozero
zero:  mov @s,#0
nozero:      inc s
          cjne s,#50h,loop1
```

3. Write a program, in 2051 assembler, that will convert an array of numbers into their ASCII equivalents.

The array of numbers is located at addresses 40h to 4Fh inclusive. The number values are restricted to 00h - 0Fh so that each can be replaced by a single ASCII character '0' - 'F', ie the ASCII equivalents of the numbers in the array.

eg if location 40h contains 06h it will be replaced by 36h which is equivalent to '6', if it contains 0Bh it will be replaced by 42h which is equivalent to 'B'.

(The equivalent values are: 0 - 9 \equiv 30h - 39h, A - F \equiv 41h - 46h)

```
; in c
; char *s,n
; for(s=0x40;s!=0x50;s++) {
;     n=*s;
;     n=n-10;
;     if(n<0) n=n+0x30+10-0x41;
;     n=n+0x41;
;     *s=n;
;
prog3:
;     s equ r0
;     mov s,#40h
loop2:mov a,@s
;     clr c
;     subb a,#10
;     jnc ge10
;     add a,#30h+10-41h
ge10: add a,#41h
;     mov @s,a
;     inc s
;     cjne s,#50h,loop2
end
```

Tutorial 8

1. Write 2051 assembly language for the following C string functions.

```
// string length
int strlen(char * s) {
    char *sc;

    for (sc = s; *sc != 0; sc++)
        /* do nothing */;
    return sc - s;
}

// concatenate two strings
char * strcat(char * dest, char * src) {
    char *tmp = dest;

    while (*dest != 0)
        dest++;
    while ((*dest++ = *src++) != 0)
        /* do nothing */;

    return tmp;
}

// compare two strings with maximum length
int strncmp(char * s, char *t, int count) {
    char r;

    while (count != 0) {
        r = *s - *t;
        if (r != 0 || *s == 0)
            break;
        s++;
        t++;
        count--;
    }

    return r;
}

/**
 * Find the first substring in a NUL terminated string
 * s1: The string to be searched
 * s2: The string to search for
 */
char * strstr(char * s1, char * s2)
{
    int l1, l2;

    l2 = strlen(s2);
    if (l2 == 0)
        return s1;
    l1 = strlen(s1);
    while (l1 >= l2) {
        l1--;
        if (!strncmp(s1, s2, l2))
            return s1;
        s1++;
    }
    return 0;
}
```

Tutorial 8 Solutions

```
$MOD2051
jmp start
; // string length
; int strlen(char * st) {
;     char *sc;
;
;     for (sc = st; *sc != 0; sc++)
;         /* do nothing */;
;     return sc - st;
;}
st equ r0
st_data 0
sc equ r1
sc_data 1
strlen:
    push sc_
    mov sc,st_
14:    mov a,@sc
        jz 15
        inc sc
        jmp 14
15:    mov a,sc
        clr c
        subb a,st
        pop sc_
        ret
; // concatenate two strings
; char * strcat(char * dest, char * src) {
;     char *tmp = dest
;
;     while (*dest!=0)
;         dest++;
;     while ((*dest++ = *src++) != 0)
;         /* do nothing */
;
;     return tmp;
;}
dest equ r0
dest_data 0
src equ r1
src_data 1
strcat:
    push dest_
    push src_
11:    mov a,@dest
        jz 12
        inc dest
        jmp 11
12:    mov a,@src
        mov @dest,a
        jz 13
        inc src
        inc dest
        jmp 12
13:    pop src_
        pop dest_
        mov a,dest
        ret
; // compare two strings with maximum length
; int strncmp(char *s,char *t,int count) {
;     char r;
;
;     while (count) {
;         r=*s-*t;
;         if (r != 0 || *s==0)
;             break;
;         s++;
;         t++;
;         count--;
;     }
;     return r;
;}
s equ r0
t equ r1
count equ r2
r equ r3
strncmp:
    push 0
    push 1
```

```

        push 2
        push 3
17:      mov a,count
        jz l6
        mov a,@s
        clr c
        subb a,@t
        mov r,a
        jnz l6
        mov a,@s
        jz l6
        inc s
        inc t
        dec count
        jmp l7
16:      mov a,r
        pop 3
        pop 2
        pop 1
        pop 0
        ret

;/**
; * Find the first substring in a NUL terminated string
; * s1: The string to be searched
; * s2: The string to search for
; */
;char * strstr(char * s1,char * s2)
;{
;    int len1, len2;
;
;    len2 = strlen(s2);
;    if (len2==0)
;        return s1;
;    len1 = strlen(s1);
;    while (len1 >= len2) {
;        len1--;
;        if (!strncmp(s1,s2,len2))
;            return s1;
;        s1++;
;    }
;    return 0;
;}

strstr:

s1 equ r0
s2 equ r1
s2_data 1
len1 equ r3
len1_data 3
len2 equ r2
len2_data 2

        push len1_
        push len2_
        push 0
        mov r0,s2_
        acall strlen
        pop 0
        mov len2,a                ; len2 = strlen(s2);
        jz rets1                  ; if (len2==0) return s1;
        acall strlen
        mov len1,a                ; len1 = strlen(s1);
112:      clr c
        mov a,len1
        subb a,len2
        jc l11                    ; while (len1 >= len2) {
        dec len1                  ;     len1--;
        acall strncmp;           ;     if (!strncmp(s1,s2,len2))
        jz rets1                  ;         return s1;
        inc s1                    ;     s1++
        jmp l12                    ; }
111:      mov a,#0                ; return 0
        jmp l10

rets1:   mov a,s1
110:     pop len2_
        pop len1_
        ret

start:   mov r0,#40h
        mov r1,#50h
        acall strstr
        mov 20h,a

end

```