EECS 281: Homework #4 Due: Tuesday, February 22, 2005

Name: _____ Email: _____

(1) Using standard C++ precision and data types, convert the following into 8051 assembler syntax and convert the the other columns into various base 2 binary complements in base 2 format.

An one's or two's complement unsigned n-bit range is 0 to $2^n - 1$ so an 8-bit unsigned char is 0 to 255.

A two's complement signed n-bit range is -2^{n-1} to $2^{n-1} - 1$ so an 8-bit signed char is -128 to 127.

A one's complement signed n-bit range is $-2^{n-1} - 1$ to $2^{n-1} - 1$ so an 8-bit signed char is -127 to 127.

A signed magnitude n-bit range is same as a one's complement signed n-bit range.

	8051 definition	big endian two's compl.	little endian two's compl.	big endian one's compl.	big endian signed magnitude
unsigned char x='a';	.byte 'a'	0110 0001	1000 0110	0110 0001	0110 0001
unsigned char x=0;	.byte 0	0000 0000	0000 0000	0000 0000	0000 0000
signed char x=−1;	.byte −1	1111 1111	1111 1111	1111 1110	1000 0001
unsigned char x=0x255;	range error 10-bit number	~~10 0101 0101~~			
unsigned char x=255;	.byte 255	1111 1111	1111 1111	1111 1111	range error 9-bit number
unsigned char x=256;	range error 9-bit number	~~1 0000 0000~~			
unsigned char x=0255;	; Octal .byte 255o	10 101 101	101 101 01	10 101 101	range error 9-bit number
signed char x=255;	range error 9-bit number	~~0 1111 1111~~	~~1111 1111 0~~	~~0 1111 1111~~	~~0 1111 1111~~
signed char x=−'a';	.byte −'a'	1001 1111	1111 1001	1001 1110	1110 0001
unsigned char x=127;	.byte 127	0111 1111	1111 1110	0111 1111	0111 1111
signed char x=127;	same as above	0111 1111	1111 1110	0111 1111	0111 1111
unsigned char x=128;	.byte 128	1000 0000	0000 0001	1000 0000	error 9-bit number
signed char x=128;	range error 9-bit number	~~0 1000 0000~~	~~0000 0001 0~~	~~0 1000 0000~~	~~0 1000 0000~~
signed char x=−128;	.byte −128	1000 0000	0000 00001	range error 9-bit number 1 0111 1111	range error 9-bit number 1 1000 0000
unsigned char x=0128;	Octal syntax error digits 0-7				
unsigned char x=−64;	unsigned error no minus				
signed char x=013;	; Octal .byte 13o	00 001 011	110 100 00	00 001 011	00 001 011
signed short x=013;	.word 13o	00000000 00 001 011	110 110 00 00000000	00000000 00 001 011	00000000 00 001 011
signed short x='a';	.word 'a'	00000000 0110 0001	1000 0110 00000000	00000000 0110 0001	00000000 0110 0001
signed short x=−'a';	.word $-'a'$	11111111 1001 1111	1111 1001 1111 1111	11111111 1001 1110	10000000 0110 0001
unsigned short x=256;	.word 256	00000001 0000 0000	0000 0000 10000000	00000001 0000 0000	00000001 0000 0000

2. Using C++ convert the following using the following values:
register unsigned char u, a=0x85, b=0xa7, c=03; register signed char s, w=0x81, x=0xa6, z=−1;
Assign the 8051 registers as follows: u=A, a=R0, b=R1, c=R2, s=R3, w=R4, x=R5; z=R6;
State if **overflow or carry** has occurred. You can double check your work by using the C compiler.

Overflow and Carry is associated with arithmetic operations: a+b, a−b, $-a$, a/b, a∗b, a<<b, a>>b.
The result of a subtract, subb, is a borrow even though it uses the same carry flag.
Overflow and Carry are not changed by non-arithmetic, boolean or data transfer operations: anl, orl, anl, mov.
Counting instructions typically used for loops and pointers, INC and DEC, do not change the carry and overflow flag.

	two's complement big endian	8051 instructions	OV	CY
a = 0x85;	10000101b	mov r0,#0x85		
b = 0xa7;	1010 0111	mov r1,#0xa7		
z = −1;	1111 1111	mov r6,#-1		
u = ∼ a;	0111 1010	mov a,r0; cpl a		

u = $-a$;	0111 1011	mov a,r0; cpl a; inc a
		mov a,r0; cpl a; add a,#1
		clr a; clr c; subb a,r0
			–
			0
			0
				–
				0
				1

u = a & b;	1000 0101	mov a,r0; anl a,r0		
u = a & w;	1000 0001	mov a,r0; anl a,r4		
u = a | b;	1010 0111	mov a,r0; orl a,r1		
u = a | (b & c);	1000 0111	mov a,r1; anl a,r2; orl a,r0;		
u = a ^ b;	0010 0010	mov a,r0; xrl a,r1		
u = a ^ 'C';	1100 0110	mov a,r0; xrl a,#'C'; mov a,r0; xrl a,#0x43		
u = a + 'C';	1100 1000	mov a,r0; add a,#'C'	0	0
u = ∼ a + 1;	0111 1011	same as previous $u = -a$	0	0
u = a − b;	1101 1110	mov a,r1; cpl a; inc a; add a,r0 mov a,r0; clr c; subb a,r1	0 0	0 1
u = a << 2;	0001 0100	mov a,r0; clr c; rlc a; clr c; rlc a mov a,r0; add a,r0; add a,acc	– 0	0 0
u = a >> 2;	0010 0001	mov a,r0; clr c; rrc a; clr c; rrc a;		0
s = ∼ w;	0111 1110	mov a,r4; cpl a;		
s = − w;	0111 1111	mov a,r4; cpl a; add a,#1	0	0
s = w + x;	0010 0111	mov a,r4; add a,r5	1	1
s = w − x;	1101 1011	mov a,r5; cpl a; inc a; add a,r4; mov r3,a mov a,r4; clr c; subb a,r5; mov r3,a	0 0	0 1
s = w ^ x;	0010 0111	mov a,r4; xrl a,r5; mov r3,a		
s = $-z$ ^ ∼ a;	0111 1011	mov a,r6; cpl a; add a,#1; mov r3,a; mov a,r0; cpl a; xrl a,r3; mov r3,a	0	0

3.Using the 8051 instruction, assemble the instruction **by pencil and paper** into hex, and then execute it showing the clock time in machine cycles:
PC points to the current instruction **before** it gets executed
Register and Flags get changed **after** the execution and placed on line below.
MOV instructions leave Flags unchanged.

Mem. Addr.	Machine instructions	Assembly	Clock Time	PC	OV	CY	AC	Reg. A	Reg. R1
0x0	0x74c8	MOV A,#0xC8	0	0x0	0	0	0	0xff	0xff
0x2	0x7988	MOV R1,#0x88	1	0x2	0	0	0	0xc8	0xff
0x4	0x29	ADD A,R1	2	0x4	0	0	0	0xc8	0x88
0x5	0xf582	MOV DPL,A	3	0x5	1	1	1	0x50	0x88
0x7	0xe4	CLR A	4	0x7	1	1	1	0x50	0x88
0x8	0x3410	ADDC A,#0x10	5	0x8	1	1	1	0x00	0x88
0xa	0xf583	MOV DPH,A	6	0xa	0	0	0	0x11	0x88
0xc		Total Time=	7	0xc	0	0	0	0x11	0x88

```
command-line>./as31 -l test2.asm

AS31 2.0b3 (beta), March 20, 2001
Please report problems to: paul@pjrc.com

Begin Pass #1
Begin Pass #2

command-line>more test2.lst

0000:                                .org    0x00
                     start:

0000: 74 C8                           mov    a,#0xc8
0002: 79 88                           mov    r1,#0x88
0004: 29                              add    a,r1
0005: F5 82                           mov    dpl,a
0007: E4                              clr    a
0008: 34 10                           addc   a,#0x10
000A: F5 83                           mov    dph,a
000C:                                .end
```