

Computer Organization

Fall 2005

Test #1 – Sept 29th

Name _____

There are 8 pages - make sure you have all of them. The last page includes an ASCII table.
Answer all questions - pay attention to the # of points for each question.
Don't leave anything blank - partial credit is always possible!

Question 1 (20 pts): Complete the following tables by filling in each blank space with the appropriate value.

Decimal	8 bit 2's complement binary	Hex
1	00000001	01
-86	10101010	AA
91	01011011	5B
-100	10011100	9C

Decimal	8 bit unsigned binary	Hex
155	10011011	9B
129	10000001	81
196	11000100	C4

Decimal	IEEE Single Precision (32 bit)		
	Sign (1)	Exponent (8)	Significand (23)
123.125	0	10000101	1110110010000000000000

123.125: 1111011.001 => 1.111011001 x 2^6
exponent is 6 -> 127+6=>133: 10000101
Mantissa is 110110010000000000

Question 2 (10 pts): We want to build a processor that will control an MP3 (music) player. The player holds 16 different songs, numbered 0-15. The operations we need the processor to understand are:

- play song number x (where x is 0-15)
- shuffle the songs (reassign song numbers randomly)
- play all songs in order starting at song number 0.
- play song number x over and over and over (forever). x can be 0-15.
- play one random song.
- increase volume
- decrease volume

The machine code instructions must be 6 bits each.

Part a (5 pts): Develop a 6-bit binary *machine code* that encodes these instructions (each instruction must be 6 bits). Provide the details of your encoding (describe your machine code here):

```
Machine Code:  x3x2x1x0 is encoding of song number
                ? means "don't care"

Play Song x:   00x3x2x1x0
Play Song x over and over: 01x3x2x1x0
Shuffle: 100???
Play all songs in order: 101???
Play one random song: 110???
Increase Volume: 1110??
Decrease Volume: 1111??
```

Part b (5pts): Using your machine code from part a, write a machine code program that corresponds to the high level description below:

```
shuffle all songs           100000
play song 13                001101
play song 7                 000111
increase volume             111000
play a random song         111000
play all songs starting at 0 101000
decrease volume            111100
play song 6 forever        010110
```

Question 3 (10 pts): The code below includes a **subtraction** operation over two 8-bit signed integers x and y . Your job is to add code that will print out whether the result z is correct or not (did the computation result in overflow?).

```
signed char x,y,z;

/* x and y acquire values somehow... */

z = x - y;

/* add code here to print "correct" or "incorrect" */

/* almost the same as checking after addition, we just
   reverse the sign of y (if we subtract a negative it's
   just like adding a positive).

if ( ((x&0x80)!=y&0x80) && ((x&0x80) != (z&0x80)))
    printf("incorrect\n");
else
    printf("correct\n");
```

Question 4 (30 pts): Short answer questions

Part a (6 pts): What is the output produced by the following program?

```
int main() {
    char *s = "CompOrg is my favorite course";
    char **p=&s;

    printf("1. %s\n",s);
    printf("2. %d\n",*s);
    printf("3. %d\n",p+1-&s);

    return(0);
}
```

Output is:

```
1. CompOrg is my favorite course
2. 67
3. 1
```

Part b (5pts): A processor is designed to hold up to 4096 bytes of memory. What is the size of a pointer (number of bits)?

bits it takes to represent any address, which is $\log_2(4096)=12$

Part c (5pts): Write the C function named `incpointer` that is called in the example below so that it adds one to a pointer to a char:

```
/* Your definition of incpointer goes here */
/* NOTE: *s++ won't work! */
void incpointer(char **s) {
    *s=*s+1;
}
int main() {
    char *s="I love pointers";
    while (*s) {
        printf("Char is %c\n",*s);
        incpointer(&s);
    }
}
```

Part d (4 pts): A processor uses 12 bit two's complement representation for signed integers.

- what is the largest positive integer that can be represented? Show your answer in decimal.

Largest number is $2^{11}-1 = 2047$

- what is the largest negative integer that can be represented? Show your answer in decimal.

$-2^{11} = -2048$

Poorly worded question, so -1 is also accepted as correct.

Part e (7 pts): What is the output produced by the following C program?

```
#include <stdio.h>

int main(int argc, char **argv) {
    char *s = "abcdefghijklmnopqrstuvwxyz";
    char c;
    int *p;
    int i;
    p = (int *)s;
    for (i=0; i<4; i++) {
        c = (* (char *) (p+i)) + 1;
        printf("%d. %c (%d)\n", i, c, c);
    }
    return(0);
}
```

Output is:

```
0. b (98)
1. f (102)
2. j (106)
3. n (110)
```

Part f (3pts): What floating point number (shown your answer in decimal) does the following IEEE 32-bit floating point number represent. Show your work!

1100 0010 1011 0001 1000 0000 0000 0000

sign is 1 (negative)

Exponent is 10000101 which is $128+4+1=133$.

$$133-127 = 6$$

Number is $1.01100011 \times 2^6 \Rightarrow 1011000.11 \Rightarrow 88.75$

Value is -88.75

- Question 5 (10 pts):** We are using a computer with the following characteristics:
- The C type int is represented using 32 bit two's complement representation.
 - The C type char is represented using 8 bit two's complement representation.
 - All pointers are 32 bits
 - All signed integers are right shifted *arithmetically*.

Consider the following variable declarations and initial code found in a single C function:

```

unsigned char ux,uy;
signed char sx, sy
char *p;
int i;

/* sx and sy have some value (we don't know what) */
ux=sx;
uy=sy;
p = &sx;
i = sx;

```

Assuming the declarations **and assignment statements shown above**, determine whether each of the expressions shown below is always true, always false, or you can't tell without knowing the actual value of the variables when the instructions are executed. I've done one for you (`ux == ux` is true for all possible values of `ux`).

Expression	always true?	always false?	can't tell?
<code>ux == ux</code>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
<code>(sx < sy) == (-sx > -sy)</code>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
<code>((sx >> 1) << 1) > sx</code>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
<code>((ux>>1)>>1) == sx>>2</code>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
<code>(unsigned char)(ux+ux) >= ux</code>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
<code>(i>>1) == (sx>>1)</code>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
<code>(ux & 0x80) == (ux < 0)</code>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
<code>(sx & 0x01) == (sx % 2)</code>	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
<code>(sx & sy) == (ux & uy)</code>	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
<code>p+3 == (char) p + (char) 3</code>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
<code>(ux & (! ux)) == 0</code>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>

Question 6 (10 pts): This question has two parts, both involve the following C statement:

$$x = (x-3) * (x-2) * (x-1)$$

Part a (5 pts): Show how this computation might look when using a processor based on a *stack-based instruction set architecture* (generic assembly is fine, with instructions like "push x", "add" and "pop tmp").

<pre>push x push 3 sub push x push 2 sub mult push x push 1 sub mult pop x</pre>	OR	<pre>push 3 push x sub push 2 push x sub mult push 1 push x sub mult pop x</pre>
--	----	--

Part b (5 pts): Show how the computation might look when using a processor based on an *accumulator based instruction set*. Generic assembly is fine here also, instructions like "add y", "load z" and "store x" are expected.

```
load x
sub 3
store tmp
load x
sub 2
store tmp1
load x
sub 1
mult tmp1
mult tmp2
store x
```

ASCII Table (decimal)

0	NUL	32	SP	64	@	96	~
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	\$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(72	H	104	h
9	HT	41)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[123	{
28	FS	60	<	92	\	124	
29	GS	61	=	93]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	DEL

Question	Possible	Score
1	20	
2	10	
3	10	
4	30	
5	10	
6	10	
Total	100	