CSE/EE 5/7385 Microcontroller Architecture and Interfacing

HOMEWORK 3

1. Assume the following values are signed ARM halfwords. Calculate their value in decimal (radix-10) and show all your work.

a) **0xFEED**
Negative value, magnitude is 0x0112+0x1=0x0113. In decimal the magnitude is $1\times16^2+1\times16^1+3\times16^0=275$. ANSWER: -275
b) **0xCAFE**
Negative value, magnitude is 0x3501+0x1=0x3502. In decimal, the magnitude is $3\times16^3+5\times16^2+2\times16^0=12{,}288+1{,}280+2=13{,}570$. ANSWER: -13,570
c) **0xACE**
Positive value (msb=0), magnitude is $10\times16^2+12\times16^1+14\times16^0 = 2560+192+14 = 2{,}766$. ANSWER: +2,766.
d) **0xFF**
Positive value (msb=0), magnitude is $15\times16^1+15\times16^0 = 255$. ANSWER: +255
2) An embedded system designer must output a one byte (1) signed magnitude value to an interface that drives a seven-segment LCD display with a plus/minus sign. Answer the following questions about signed magnitude values.

a) What is the maximum positive value in both decimal and hexadecimal?
Hex: 0x7F Decimal: $7\times16^1+15\times16^0=+127$
b) What is the minimum negative value in both decimal and hexadecimal?
Hex: 0xFF Decimal: $-1\times(7\times16^1+15\times16^0)=1127$
c) What decimal value does **0x80** correspond to?
Corresponds to a negative zero (which doesn't make sense mathematically). Answer is zero (0). Would Display "-0".
d) What hexadecimal value should be output to the LCD display if the decimal value 0 (zero) is to be displayed?
It should be 0x00 and NOT 0x80 since 0x80 would cause the negative sign to be displayed and negative zero does not make sense.
3) Using the smallest size among (BYTE, HALFWORD, WORD), give the hexadecimal value for the following decimal values. Assume that 2's complement is used for signed values. For full credit, show your calculations, do not merely use a calculator and give the result.

a) -17,635
This requires two bytes and is thus a HALFWORD. The magnitude is 0x44E3, thus the 2's complement form becomes 0xBB1C+0x1=0xBB1D. ANSWER: HALFWORD-0xBB1D.
b) -47
This a single BYTE. The magnitude is 0x2F, thus the 2's complement is 0xD0+0x1=0xD1. ANSWER: BYTE-0xD1.

c) 238

This has magnitude 0xEE, however since the MSb is set, it would be interpreted as a negative value if it were stored in a BYTE and thus be incorrect. ANSWER: HALFWORD-0xEE.

d) -128

This value has magnitude 0x80, however it must undergo 2's complementation to indicate negation. Therefore, the magnitude is given as a HALFWORD 0x0080 and is complemented as 0xFF7F+0x1=0xFF80. ANSWER: HALFWORD-0xFF80.

4) Translate the following string given between the double quotes into it's 8-bit ASCII equivalent. Give your answer as a set of hexadecimal bytes and include the spaces.

"**HW 3 Question**"

The ASCII standard requires 7-bits for each character, however, it is commonly used by setting the MSb to 0 and occasionally by using the MSb as an odd or even parity bit. In this case, 8-bits are requested, but no parity is mentioned, thus the MSb is reset. Using the ASCII table below, the translation is:

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

"H"-0x48, "W"-0x57, Space-0x20, "3"-0x33, Space-0x20, "Q"-0x51, "u"-0x75, "e"-0x65, "s"-0x73, "t"-0x74, "i"-0x69, "o"-0x6F, "n"-0x6E

In block form (from left to right)

ANSWER: 48 57 20 33 20 51 75 65 73 74 69 6F 6E

5) What constant would be loaded in register **r0** after the following instructions execute. Give your answer in BOTH hexadecimal and decimal. Assume the constants represent signed 2's complement values

a) **mov r0, #0x9D, 5**

0x0000009D rotated right by 5 bits becomes 0xA8000004. Since the MSb=1, this is a negative value. The magnitude is then 0x57FFFFFB+0x1=0x57FFFFFC.
Converting this magnitude to decimal:
$5\times16^7+7\times16^6+15\times16^5+15\times16^4+15\times16^3+15\times16^2+15\times16^1+12\times16^0$
=1,342,177,280+117,440,512+15,728,640+983,040+61,440+3,840+240+12
=1,476,395,004
ANSWER: -1,476,395,004 0xA8000004

b) **mov r0, #0x53, 30**

0x00000053 is rotated right by 30 bits which is the same as rotating left by 32-30=2 bits. Thus, the rotated value is 0x0000014C. This is a positive value since MSb=0. The magnitude is $1\times16^3+4\times16^2+12\times16^0$=4,096+1,024+12=5,132.
ANSWER: +5,132 0x0000014C

c) **mvn r0, #53**

The second operand is specified as a decimal value, so it must be converted to hexadecimal and then negated. The hexadecimal value is 0x00000035 and the negated form is 0xFFFFFFCA. Since the MSb=1, this is interpreted as a negative value. The magnitude of the negative value is 0x00000035+0x1=0x00000036. In decimal, this becomes $3\times16^1+6\times16^0$=49.
ANSWER: -49 0xFFFFFFCA

d) **mvn r0, #255, 30**

The second operand is given in decimal, so it must be converted to its hexadecimal form so it can be negated. 255=0x000000FF and when negated becomes 0xFFFFFF00. Rotating right by 30 bits is equivalent to rotating left by 2 bits, therefore the rotated value is 0x FFFFFF2B. Since the MSb=1, this represents a negative value. The magnitude is 0x000000D4+0x1=0x000000D5. In decimal, this magnitude is $13\times16^1+5\times16^0$=213.
ANSWER: -213 0xFFFFFF2B

6) Using any combination of **mov**, **mvn**, with or without rotates, or shifts, give a single instruction that will cause the following constants to be loaded into register **r1**.

a) **0xFF**

There are several correct answers. Two of these are:
mov r1, 0xFF, or **mvn r1, 0xFFFFFF00**
ANSWER: **mov r1, 0xFF**

b) **0xC400**

There are several correct answers. One of these is:

```
mov  r1,  0xC4,  24
```

c) **0x7D8**

There are several correct answers. One of these is:

```
mov  r1,  0xFB,  29
```

d) **0x17400**

There are several correct answers. One of these is:

```
mov  r1,  0x5D,  20
```

e) **0x1980**

There are several correct answers. One of these is:

```
mov  r1,  0x5D,  20
```

f) **0xA50000**

There are several correct answers. One of these is:

```
mov  r1,  0xA5,  8
```

7) Give the single instruction that will load the following constants into register r0. You may ONLY use a literal pool if that is the only possible way to load the instruction.

a) **0xFFFFFFFF**

There are several correct answers. One of these is:

```
mvn  r0,  0x0
```

b) **0x12340000**

There are several correct answers. One of these is:

```
ldr  r0,  =0x1234
```

c) **0xFFFFFFFE**

There are several correct answers. One of these is:

```
mvn  r0,  0x1
```

d) **0x88888888**

There are several correct answers. One of these is:

```
ldr  r0,  =0x88888888
```

8) Give the total number of **bits** (as a decimal, base-10 number) for the following. Your answer should be in total number of bits in the form $\mathbf{A}\times 2^{\mathbf{B}}$ where **A** is NOT a factor of 2 (two). Do not use Kilo-, Mega-, or Giga-. An example is 6 Kbits = 3×2^{11} so **A**=3 and B=**11**.

ARM word: _A=1_B=5___ 32 Mbits: _A=1_B=25_____ Byte: _A=1_B=3__

16 GBytes: _A=1 B=37_ ARM halfword: _A=1 B=4_ 48 KBytes: __A=3_B=17__

9) The diagram below depicts a portion of the ARM memory before the instruction executes. Assume that the processor is operating in ***little endian*** mode, is executing the sequence of instructions below, and that it DOES support unaligned accesses.

ADDRESS	DATA
0x8000	0xA1
0x8001	0x33
0x8002	0x2C
0x8003	0xFF
0x8004	0x00
0x8005	0x7D
0x8006	0x7E
0x8007	0x7F
0x8008	0x37
0x8009	0x5B

```
mov  r0, #0x8004          ;r0 <- 0x00008004
ldr  r1, [r0, #-1]        ;r1 <- 0xA1332CFF
ldr  r2, [r0]             ;r2 <- 0x332CFF00
ldr  r3, =0xFFFFFFFF      ;r3 <- 0xFFFFFFFF
and  r1, r1, #0xff        ;r1 <- 0x000000FF
and  r2, r2, #0xff        ;r2 <- 0x00000000
cmp  r2, #255       ;Instruction A    r2-0xff=0x=0xffffff01
cmp  r1, #255       ;Instruction B    r1-0xff=0x00000000
cmp  r3, r1         ;Instruction C    r3-r1=0xffffff00
cmp  r1, #0         ;Instruction D    r1-0x0=0x000000ff
```

a) Give the flag contents immediately after `Instruction A` executes:

N = _1__ C = _0__ Z = _0__ V = _0__

b) Give the flag contents immediately after `Instruction B` executes:

N = _0__ C = _1__ Z = _1__ V = _0__

c) Give the flag contents immediately after `Instruction C` executes:

N = _1__ C = _1__ Z = _0__ V = _0__

d) Give the flag contents immediately after `Instruction D` executes:

N = _0__ C = _1__ Z = _0__ V = _0__