COMP9032 Lab 1

Sept. 2023

For this lab, you are required to work solo, and your work is assessed **individually**.

1. Objectives

In this lab, you will learn:

- AVR instructions, and
- basic assembly programming.

2. Programming Style

The general practice, when you write an assembly program, is to maintain the readability and consistency of your code. For this reason, you are encouraged to adopt the following rules:

- Starting each source code file with a heading that includes:
  - your name so that it is easy to see who is responsible for the file, the date of last modification and a version number, and
  - **a description of what the program does, possibly with a pseudo-code for a high-level abstraction.**
- Including appropriate comments that explain the “why", not just the “how" of the program throughout the source code.
- Using a sensible layout for your code - to make it easy to see the code structure, instructions, and any labels.

3. Tasks

There are three tasks in this lab.

3.1 Task 1 (15 marks, **due your lab session in Week 2**)

Write an assembly program that converts a signed number in a register into the decimal and store the decimal number in a group of registers where each register holds the ASCII value of the sign or the digit of the decimal. An example is shown in Figure 1, where the signed binary 11111011 in R3 is converted to decimal -5 which is stored in register pair R4:R5 in the ASCII format (See the ASCII table given at the end of this document).

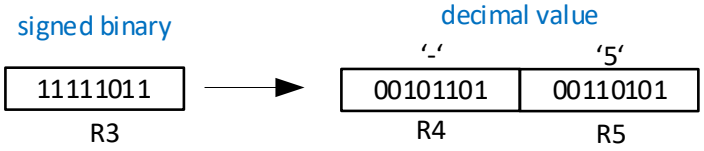

Figure 1: Signed binary 11111011 is converted to decimal and stored in R4:R5 in ASCII

Here we assume the input value will be manually set in the register before execution (See explanation on page 5 of Lab 0 on how to set a register value).

3.2 Task 2 (15 marks, **due your lab session in Week 3**)

Write an assembly program to calculate $a\sqrt{a}$, where a is a one-byte unsigned number and the result is rounded to integer, e.g. $5\sqrt{5} = 11.18 = 11$.

Here we assume value a is stored in a register, which will be manually set to the value before execution and the result will be saved in another register.

3.3 Task 3 (15 marks, **due your lab session in Week 3**)

The greatest common divisor (GCD) of two integers can be calculated in a way as given in a C-like pseudo code shown in Figure 2. Based on this calculation approach, write an assembly code to get the GCD of an array of 8-bit unsigned integers. The size of the array is less than 10 and is stored in register R0. The elements of the array are stored consecutively from register R1 up to R9. Both the array size and the array are manually set before execution.

```
/* below is only part of C function
char a, b;                    /* 8-bit unsigned integer

       while (a!=b)
       {                      /* Assume a, b > 0 */
              if (a>b)
                              a = a - b;
              else
                              b = b - a;
       }
/* a and b both hold the result */
```

Figure 2: Pseudo Code GCD

For this task, you are required to use a macro to improve your code. The concept of macro is discussed in Week 2.

NOTE:

- You can put your code for each task in the same project in microchip Studio for this lab. Run the program for each task by setting it as the entry file, which has been explained in Lab 0.
- All your programs should **be well commented on and easy to read**. Up to 10% marks will be deducted for each program without proper and sufficient comments.

Appendix: ASCII Table

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