COMP9032 Lab 1

Sept. 2019

1. Objectives

In this lab, you will learn

- AVR instructions
- 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, especially for this course:

- 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
    - **the 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 structures, instructions and any labels.

3. Tasks

This lab consists of two tasks.

3.1 Task 1 (10 marks, due in Week 2)

Write an assembly code that converts a given data of the string type into a binary number. The string can be a decimal or a hexadecimal number.

For simplicity, here we assume the string data only contains two digits. Each digit in the string is represented by an ASCII code. For example, the decimal number string “78” is represented by 0x37 for ‘7’ and 0x38 for ‘8’. (See the ASCII table shown at the end of this document).

We also assume that each ASCII code is stored in a register and the result of the conversion is also saved in a register. Since ASCII only uses 7 bits, you can use the leftmost bit to indicate the string is for decimal or hexadecimal, for example 0 for decimal and 1 for hexadecimal, as shown in the example shown in Figure 1.

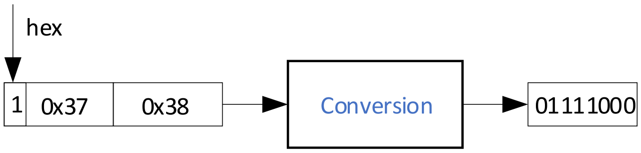


Figure 1

Note, do not initialize the registers within your code. You should be able to initialize the values in AVR Studio (refer to lab0 for the description of changing register value).

Assemble and run your program in AVR Studio and show your working program to the lab tutor. Remember to save and delete your work on the lab machine before you leave the laboratory.

3.2 Task 2 (10 marks, due in Week 3)

Manually convert the function described by gcd.c below (i.e. calculating the greatest common divisor of two numbers) into AVR assembly code (*gcd.asm)*. Assume the size of an integer is 2 bytes. Try to reduce your code size and then the execution time (extra 2 bonus marks will be granted for the best solution, to be announced in the following week).

```
int main(void)
{
            int a, b;                       /* Initialized elsewhere */

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

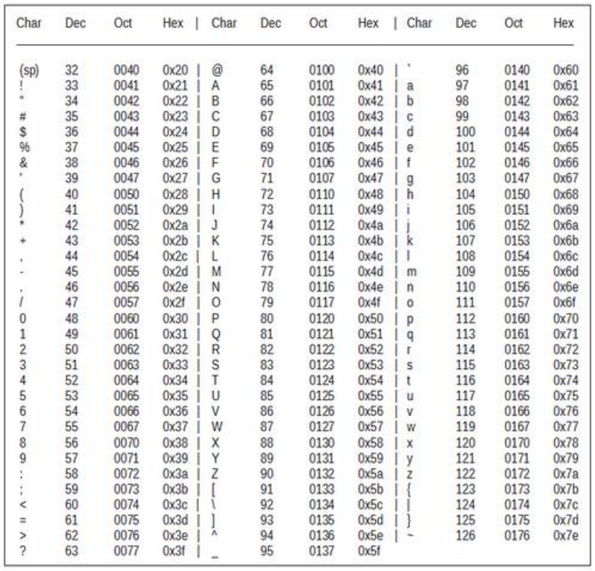
Figure 2: Program gcd.c

Assemble and run your program in AVR Studio and show your working program to the lab tutor. Remember to save and delete your work before you leave the laboratory.

NOTE: All your programs should be well commented and easy to read. Up to 1 mark will be deducted for each program without proper and sufficient comments.

Please have your work marked in your lab class of the due week.

Appendix: ASCII Table



Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
?	63	0077	0x3f	_	95	0137	0x5f				