Lab 6 Subroutines

I. Subroutines

In first year programming language courses such as csc 110, csc 111, we wrote many functions. In assembly language, the term procedure, function, or subroutine can be used interchangeably. A function call implies a transfer (branch, jump) to an address representing the entry point of the function, causing execution of the body of the function. When the function is finished, another transfer occurs to resume execution at the statement following the call. The first transfer is the function call (for invocation), the second transfer is the return. Together, this constitutes the processor's call-return mechanism.

Example: passing two parameters by values. Download params.asm (kindly provided by Mr. Jason Corless).
The diagram of the internal memory of the SRAM when "lds ZH, Z+9" is executed:

Address	content	details	notes
0x0000 ~ 0x01FF		General Purpose Registers and I/O Registers	
0x0200			.DSEG
	……..		
		<Z-and SP	
	r1	saved register	Callee pushes those registers onto the stack in the subroutine to preserve the values in those registers. Why?
	r0	saved register	
	r31	saved register	
	r30	saved register	
	ret	return address	Assembler pushes the return address onto the stack automatically when "call do_something" is executed. 22 bits address -> 3 bytes are needed.
	ret	return address	
	ret	return address	
	0xCC	parameter (Z + 8)	Caller pushes the values onto the stack in the "main" in the example.
0x21FF	0xEE	parameter (Z + 9)	

Build params.asm and run the code, observe the contents of the registers SP, Registers Z, r16, r1 and r0.

II. Exercises: download lab6.asm, read void strcpy (src, dest) subroutine. Understand it, reconstruct the stack frame. Implement unsigned int upperCaseCount (str) subroutine (count the number of upper case letters) using the two subroutines we have just learnt as examples.

To test if each character is in upper case, use the ASCII code table below. ACSII stands for American Standard Code for Information Interchange, is a character encoding standard for electronic communication.

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

Read the code in lab6.asm. Write on a piece of paper the stack frame for strcpy.

Address	content	details	notes
	……		
		< SP	
		……	Callee (subroutine) pushes those registers onto the stack in order to preserve the values in the register.
	ret	return address	Assembler pushes the return address onto the stack automatically when “call strcpy” is executed.
	ret	return address	
	ret	return address	
	…..		Caller pushes parameters on to the stack: the memory address of the destination string.
			Caller pushes parameters on to the stack: the memory address of the source string.
0x21FF			

Design the stack frame for upperCaseCount (str)

Address	content	details	notes
	……..		
		< SP	
		……	Callee (subroutine) pushes those registers onto the stack in order to preserve the values in the register.
	ret	return address	Assembler pushes the return address onto the stack automatically when “call upperCaseCount” is executed.
	ret	return address	
	ret	return address	
	…..		Caller pushes parameter on to the stack: the memory address of the source string.
0x21FF			