Lab 6 Subroutines

Submit lab6.asm at the end of your lab.

I. Subroutines

In lab 4, we learned how to write a simple subroutine, that is, a subroutine without parameter passing nor a returned value. For example, in C or Java, the prototype of a simple subroutine looks like this:

```
void delay()
```

In today's lab, we are going to learn how to write subroutines with parameter(s) and returned value.

Example 1: passing two parameters by value using a stack on SRAM (the data memory), the returned value is saved in register pair sumH:sumL (r19:r18). In C or Java, the equivalent subroutine looks like this:

```
int add_num (int n1, int n2)
```

In assembly language, the parameters n1 and n2 are pushed onto the stack before calling the subroutine add_num. **Download params.asm** and add it to your project.

The diagram of the internal memory of the SRAM when "ldd n1, Z+9" is executed:

Address	content	details	notes
0x0000 ~ 0x001F		General Purpose Registers	
0x0020 ~ 0x005F		64 I/O Registers	
0x0060 ~ 0x01FF		416 extended I/O Registers	
0x0000 ~ 0x01FF			
……			
0x0200			.DSEG
……			
0x21F6		<-SP	"0x21F6" is stored in SP (**S**tack **P**ointer)
0x21F7	n2 (r1)	saved register	In the subroutine "add_num", the callee pushes those registers which will be used onto the stack to protect them from being changed.
0x21F8	n1 (r0)	saved register	
0x21F9	ZH (r31)	saved register	
0x21FA	ZL (r30)	saved register	
0x21FB	ret	return address	The return address (the address of the next command) is pushed onto the stack automatically when "call add_num" is executed.
0x21FC	ret	return address	
0x21FD	ret	return address	
0x21FE	0xCC	parameter (Z + 8)	In the main, the caller pushes the arguments onto the stack.
0x21FF	0xEE	parameter (Z + 9)	

Three bytes are used for the return address since size of the flash memory is 256KB (2^{18} bytes) (refer to page 7, Table 2-1 of the datasheet).

Build params.asm and run the code, press F11 and observe the contents of the registers: SP, Registers Z, r16, r1 and r0, r19:r18

Example 2: passing parameter by reference using a stack on SRAM (the data memory), no returned value.

Download lab6.asm, read void strcpy (src, dest) subroutine. Understand it, reconstruct the stack frame. Implement unsigned int strlength (str) subroutine using the two subroutines we have just learnt as examples.

Read the code in lab6.asm. Download stackFramePractice.docx and write the stack frame for strcpy using the example on the previous page.

Address	content	details	notes
	……..		
		<- SP	**Stack Pointer**
		……	In the subroutine, the callee pushes those registers onto the stack in order to preserve the values in the registers.
	ret	return address	The return address (the address of the next command) is pushed onto the stack automatically when "call strcpy" is executed.
	ret	return address	
	ret	return address	
	…..		In the main, the caller pushes parameters on to the stack: the memory address of the destination string.
			In the main, the caller pushes parameters on to the stack: the memory address of the source string.
0x21FF			

Design the stack frame for strlength(String str):

Address	content	details	notes
	……..		
		<- SP	**Stack Pointer**
		……	In the subroutine, the callee pushes those registers onto the stack in order to preserve the values in the registers.
	ret	return address	The return address (the address of the next command) is pushed onto the stack automatically when "call strlength" is executed.
	ret	return address	
	ret	return address	
	…..		In the main, the caller pushes parameter on to the stack: the memory address of the source string.
0x21FF			

Submit lab6.asm at the end of your lab.