Read the code in lab6.asm. Write the stack frame for strcpy using the example on page 2 of lab6.pdf. You do not need to submit this document.

strcpy

Address	content	details	notes
……..			
Y → 0x21F1		<- SP	**Stack Pointer**
F2	r23	……	In the subroutine, the callee pushes those registers onto the stack in order to preserve the values in the registers.
F3	r27		
F4	r26		
F5	r28		
F6	r29		
F7	r31		
0x21F8	r30	Saved reg.	
0x21F9	ret	return address	The return address (the address of the next command) is pushed onto the stack automatically when "call strcpy" is executed.
0x21FA	ret	return address	
0x21FB	ret	return address	
	…..		In the main, the caller pushes parameters on to the stack: the memory address of the destination string.
X → dest 0x21FC	0x00	parmt 11	
0x21FD	0x02	parmt 12	In the main, the caller pushes parameters on to the stack: the memory address of the source string.
Z → src 0x21FE	0x5C	parmt 13	
0x21FF	0x00	parmt 14	

strlength

SP → 0x21F4		
F5	r27	
F6	r26	
F7	r28	
F8	r29	
F9	r31	
FA	r30	
FB	ret	
FC	ret	
FD	ret	
FE	0x00	pt 10
0x21FF	0x02	pt 11

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			