Juan Madrigal
CST 337
Lab 4 Report

3.

a) Did resetting the device change any of the RAM?
   i) **No**

b) Record and report the virtual addresses of the global variables and their initial values:
   i) **const char a[]**
      1) **Virtual Address: 0x9D000D38**
      2) **Initial value: "CST 337 Lab 4\0"**
   ii) **char b[]**
      1) **Virtual address: 0x800002B4**
      2) **Initial value: "UUUUUUUUUUUUUUUUUUUUUUU"**
   iii) **const char c[]**
      1) **Virtual address: 0x9D000D48**
      2) **Initial value: "Initialized Constant String\0"**
   iv) **char d[100]**
      1) **Virtual address: 0x800002CC**
      2) **Initial value: "UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU UUUUUUUUUUU"**
   v) **unsigned int gc**
      1) **Virtual address: 0x800002AC**
      2) **Initial value: 0x55555555**
   vi) **unsigned int gd**
      1) **Virtual address: 0x800002B0**
      2) **Initial value: 0x055555555**
   vii) **unsigned int ge**
      1) **Virtual address: 0x80000280**
      2) **Initial value: 0x55555555**

c) In what memory segment do the addresses of the global constants and global variables fall under?
   i) **Non constants are in Data RAM and KSEG0**
   ii) **Constants are in Program Flash and KSEG0**

4.

a) Initial values of
   i) **SP: 0x00000000**
   ii) **FP: 0x00000000**
   iii) **GP: 0x00000000**

b) Check address of pc after reset is 0xBFC00000

i) What segment of memory is this address in?
   1) **Boot Flash and KSEG1**

ii) What does table 6.8 in the MIPS Volume III manual say this is the address of?
   1) **The Exception Vector Base Addresses**

5.

a) After stepping into 13 times record the final value of:
   - **i) SP: 0x8007FFF8**
   - **ii) GP: 0x80008270**

b) What segment of memory is each of these registers pointing to?
   - **i) RAM and KSEG0**

6.

a) Run to main
   - i) Did sp or gp change?
     1) **SP changed to 0x8007FFE0**
     2) **GP didn’t change**
   - ii) Which global values changed?
     1) **b = "Initialized Global Var\0"**
     2) **d = "Initialized String Array\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"**
     3) **gc = 0x00000000**
     4) **gd = 0x00000000**
     5) **ge = 0x00000045**
   - iii) What value was gd initialized to?
     1) **0x00000000**
   - iv) What caused all these variables to change?
     1) **initialize stack pointer and heap step in the start up code caused sp and gp to change.**
     2) **And the on reset routine initializes the global variables.**

7.

a) What is common about the declarations of these local variables that already have their initial values?
   - **i) They are static type and the other one is char type.**

b) Of these variables, some fall in KSEG0 Program Flash, and some in KSEG0 Data RAM. What is different about the type of declarations of the variables that fall in Program Flash, and those that fall in Data RAM?
   - **i) The program flash variables are constant and the data ram variables are not constant.**

c) When do you think each of the values were initialized?

i) **I think the non constant data variables are initialized in step 10 of the runtime startup code. I think the constant program flash variables are initialized at step 11 of the runtime startup code.**

9.

a) What registers have values close to these addresses?
   - i) **SP and FP.**

b) How would we commonly refer to the section of RAM that contains these addresses?
   - i) **The stack contains local variables.**

c) It may surprise you that the non-static const local variables reside in RAM. So, how can constant local variables be constant?
   - i) **They are in the stack, and we can be sure they are safe to be constant because they are thread safe.**

10.

a) Record where the data used to initialize ic is stored.
   - i) **It's in the program because the value is loaded immediately.**

b) In what memory segment is this initializer data?
   - i) **KSEG0 and Program Flash**

11.

a) At what address is the first byte of the string data used to initialize la stored?
   - i) **0x9D000D64**

b) In what memory segment is this?
   - i) **KSEG0 and Program Flash**

c) Is local string initialization this way a good thing?
   - i) **Yes because in terms of speed it is fast by saving the string by a word at a time.**

d) Would it be better for us to have used strcpy() to initialize la?
   - i) **No because strcpy() is twice as many instructions because it saves the string by a byte at a time.**

12.

a) KSEG0 addresses for initializers b[], d[], ge.
   - i) **b[] = 0x1D000254**
   - ii) **d[] = 0x1D00026C**
   - iii) **ge = 0x1D0002E8**

b) In what type of memory do these initializers fall under?
   - i) **Program Flash in Physical Memory**

c) Why do you suppose that the addresses for the global strings a[], b[] initializer, c[] and d[] initializer are not sequential?
   - i) **Because a[] and c[] are in virtual memory map and b[] d[] initializers are in physical memory map.**

d) Which approach is more memory efficient for strings like a[], and b[]; to use the const declaration in a[] or not as in b[]?
   i) **It's more memory efficient for strings to not use const declaration as in b[].**

e) Why is it more efficient?
   i) **Because the const declaration creates a copy of the const value which takes more memory.**

13.

a) What is the problem with this strcpy?
   i) **There is no control of changing the size of the string or how much of the string to copy over.**

b) Where do the extra data go?
   i) **The extra data is over writing data in the next variable memory space. In this case it is over writing d[].**

c) What determines the end of a string as far as strcpy() is concerned?
   i) **Null**

d) How does the debugger determine how long strings are for purposes of displaying them in the watch window?
   i) **The initialized size of the variable which is determined by the initial string assigned to it.**

e) What do you think would happen if we added a strcpy(d, b) following the last strcpy(b, c)?
   i) **The whole string in c[] will be copied over but the remaining previous data of b[] will stay since it wasn't overwritten since the copy only wrote 23 bytes of data.**

f) Explain what happened and why it happened?
   i) **What I expected did not happen. What happened was that b[] was continuously copied over to d[] and filled its memory space because b[] did not have a null character in the end. But because b[] is next to d[] in memory b[] is writing its own read that it is continuously stuck in.**

14.

a) Record the starting and ending addresses of main from the disassembly window.
   i) **Start: 0x9D000648**
   ii) **End: 0x9D0008A0**

15.

	section	variable	address
		init b	0x1D000254
		init d	0x1D00026C
		init ge	0x1D0002E8
		init la	0x1D000D64
RAM			
	.sdata		0x80000280
		ge	0x80000280
		isf	0x80000284
	.sbss		0x800002AC
		gc	0x800002AC
		gd	0x800002B0
	.data		0x800002B4
		b	0x800002B4
		d	0x800002CC
	heap		0x80000338
	stack		0x80000360
Flash			
	.text.main_entry		0x9D000110
	.app_excpt		0x9D000180
	.vectors		0x9D000200
	.dinit		0x9D000248
		start main	0x9D000648
		end main	0x9D0008A0
	.rodata		0x9D000D38
		a	0x9D000D38
		c	0x9D000D48
		Lcc	0x9D000DA8
		iscf	0x9D000DB0
	.reset		0xBFC00000
	.bev_excpt		0xBFC00380

a)

16.

a) At what address is the strcpy() routine stored in memory?
    i) **0x9D0008A4**

17.

a) What is the purpose of each of the .reset, .bev_excpt, and .app_excpt sections?
    i) .reset
        1) **Reset exception vector address**
    ii) .bev_excpt
        1) **STATUS BEV Interrupt Exception vector address**
    iii) .app_excpt
        1) **General exception vector address**

18.

a) Why does the bottom of the stack move up?
    i) **Because the heap space increased so it moved the stack up more.**
b) Why do the addresses of a and c change after making this code change?
    i) **Because the program flash memory has been moved up as well from the heap space increase.**