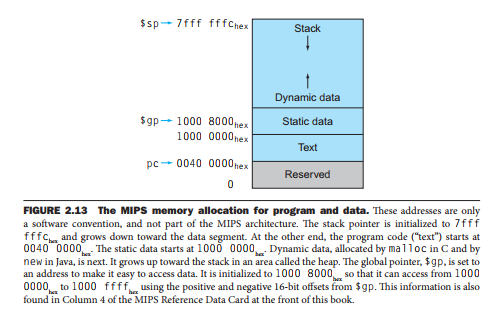
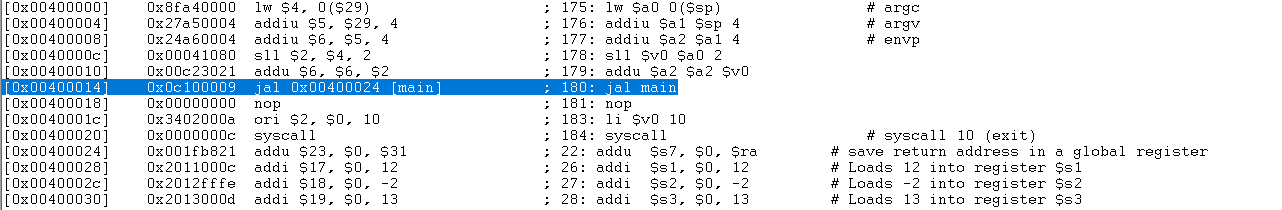
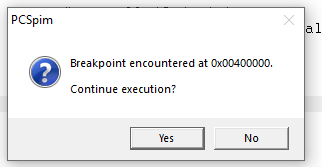
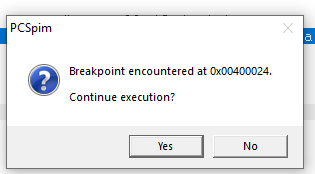
**Tutorial 3**

**a) When the loader loading user’s code into PCSpim, the system should have some way to effectively  
determine the 1st instruction of user’s code. Think about the mechanism for PCSpim to locate the** 1stinstruction **of user’s code based on the programs’ memory image. Note that PCSpim doesn’t have preknowledge of user’s source code. It is also assumed that you understand the entry point concept from  
learning Programming Fundamentals.**  
  
The instructions that are located in the code are split up into segments; data segment, text segment; when PCspim is loaded with the instructions; the first instruction is located where the text segment is. It will recognize its location and then allocate memory for those instructions into memory within the MIPS format. In reality this code would be in an object file; and a linker will get all the objects and then dedicate them addresses according to the following format:   
  
***Note: In this case we have noticed that the text segment starts at address 0040 0000 (hex).***  
  


**Question: Describe how would you recognize the right address at** the first instruction **of user’s code  
loaded into PCSpim. Illustrate to your tutor the value of the address you would enter for setting a  
breakpoint to stop user’s program execution at** the first instruction**. What message would you see when  
starting the program after setting such a breakpoint? [**0.4 marks**]**  
  
*User Text Segment [00400000]-[00440000] – the memory of the text segment is between the value specified; there is a set of instructions that operate before the program starts to execute the instructions which are* ***contained in the object file.*** *This appears to be preliminary code which jumps to the address listed in main (using jump and link instruction).*  *******I have some slight concerns about the definition of the first instruction that the tutor is asking; and therefore; there are two possible outcomes. Either the instruction that starts at memory location:**1. 0x00400000  
2. 0x00400024*

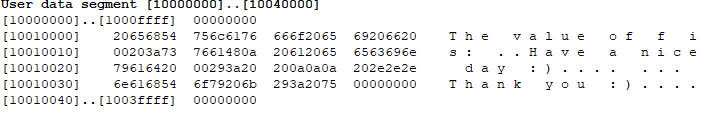
*Note: Look at hand at memory location 00400024*

When QtSpim is about to execute the instruction where there is a breakpoint, it asks for continue, single stepping or abort.  
  
**b) What would you expect to see in a memory location which was loaded (or initialised) with a single ASCII  
character “H”, “a”, and “v” respectively? Give answer in two formats: in HEX and in binary. [0.6 marks]**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **“H”** | **“a”** | **“v”** |
| **Hex** | *48* | *61* | *76* |
| **Binary** | *0100 1000* | *0110 0001* | *0111 0110* |

**2. Use a text editor to modify the *simplecalc.s* program (you completed in lab 2) to perform the following calculations: (g + h) – (i – j + k), and to output (print) the result on the screen. As usually, save your modified program with a different name.**  
[Completed look at folder for file].

**Question (0.5): load *simplecalc.s* into PCSpim, insert a breakpoint after all initial values were set, and before the calculations start. Run the program to the breakpoint, and when it stops, set a new value (any number) in one of the registers holding your initial values. Complete the run, and check the result in the console window. Demonstrate to the tutor your program, how it runs, how you can set breakpoints, and how you can modify registers midway through your program execution. Note: in PCSpim, to change register values, use ‘Simulator’ -> ‘Set Value’ function.**  
  
[completed changed value of $s1 to 25, and the value of f changed to 10]

**3. Still with *simplecalc.s* in PCSpim. In the data segment of PCSpim, identify the memory locations  
holding all characters of the message: “have a nice day :)”. There are a couple of ways to find the  
memory addresses. By the way, you may need to use ASCII table “ascii\_chart.pdf” for this question.**  
  


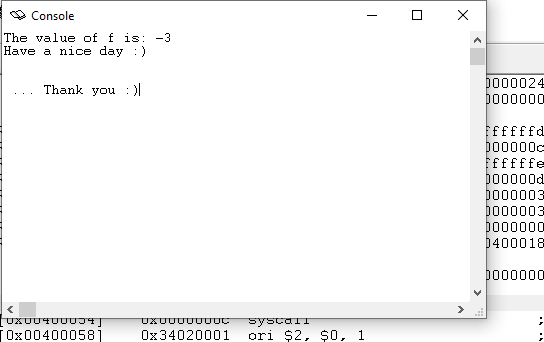
10010015 to 10010028(+the dot for asciiz) is where it is held in memory in QTspim.

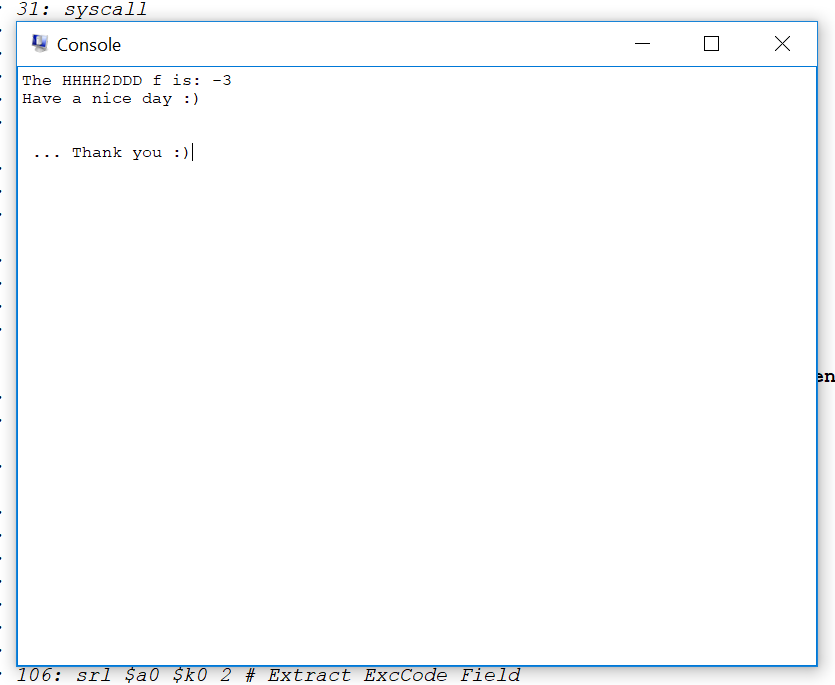
**Question (0.5):** Explain how ASCII characters are placed in memory, illustrate your explanation with a  
hand drawing. Be able to show how data is placed in memory on PCSpim/QtSpim screen. **Hints:** ASCII  
characters are packed in word blocks and placed in memory from lower memory address to higher  
memory address. The addresses are shown as hex values. The address for each word block (4 characters  
packed in it) is indicated or calculated, the address of each char in the word can be calculated by adding  
its offset.

***The string values are put into the text file in the data segment; and when QTspim runs the program; it stores those values between the memory positions: 100000000 and 100040000. The asscii characters are interpreted from hexi-decimal, binary or decimal characters; and they are representative of symbols (refer ASCII table); the machine interprets this information and holds the value of the ASCII characters into memory.***

**Question (0.5): In this exercise, you are going to change data in memory on PCSpim directly on the fly when running at a certain breakpoint (rather than changing its definition). Now insert a breakpoint right before the program prints on the screen the message: “\nHave a nice day :)”. If you are not certain where to insert the breakpoint, use single step to find out. Run the program and stop it at the breakpoint; change the four characters of the message from “*Have*” to: “*-not*” (don’t change other characters); complete the run; check the result in the console window. Hints: The PCSpim interface is read-only and doesn’t support data modification through its interface directly. Note: in PCSpim, to change data at a certain memory address, use ‘Simulator’ -> ‘Set Value’ function.**PCspim: Nothing changed, the same thing was printed.

Qtspim: The value changed





4. Analyse provided code *temperature.s* **Question (0.3): insert missing code, and fill in missing comments (lines marked #\_\_??). The program needs to run correctly to be assessable. Note: this simple program does not have to check for incorrect characters, or out of range number entries.**

Completed.

**Question (0.2): what formula was used to convert temperature from Fahrenheit to Celsius?**

Formula is Tc = (T-32)\*5/9