**Take Home Test 1 CSC 342**

Danish Faruqi

April 5, 2021

Start Time and Date: April 2, 2021 1:53pm

End Time and Date: asa

***“I will neither give nor receive unauthorizes assistance on this TEST. I will use only one computing device to perform this TEST. I will not use cell while performing this TEST” - Danish Faruqi***

**Table of Contents**

1. **[Title Page](#Title_Page)**
2. [**Table of Contents**](#Table_of_Contents)
3. [**Objective**](#Objective)
4. [**Part 1 - MIPS**](#MIPS)
   1. [2-2\_1.asm](#MIPS_2_2_1)
   2. [2-2\_2.asm](#MIPS_2_2_2)
   3. 2-3\_1.asm
   4. 2-3\_2.asm
   5. 2-5\_2.asm
   6. 2-6\_1.asm
   7. 2-7\_1.asm
   8. 2-7\_2.asm
   9. 2-8\_1.asm
   10. natural\_generator.asm
   11. main\_myadd.asm
5. **Part 2 - x86 Intel on Windows 32-bit**
   1. 2-2\_1.c
   2. 2-2\_2.c
   3. 2-3\_1.c
   4. 2-3\_2.c
   5. 2-5\_1.c
   6. 2-6\_1.c
   7. 2-7\_1.c
   8. 2-7\_2.c
   9. 2-8\_1.c
   10. natural\_generator.c
   11. main\_myadd.c
6. **Part 3 – LINUX, gcc, gdb 64bit on Intel x86-64 ISA**
   1. 2-2\_1.c
   2. 2-2\_2.c
   3. 2-3\_1.c
   4. 2-3\_2.c
   5. 2-5\_1.c
   6. 2-6\_1.c
   7. 2-7\_1.c
   8. 2-7\_2.c
   9. 2-8\_1.c
   10. natural\_generator.c
   11. main\_myadd.c
7. **Conclusion**

**Objective**

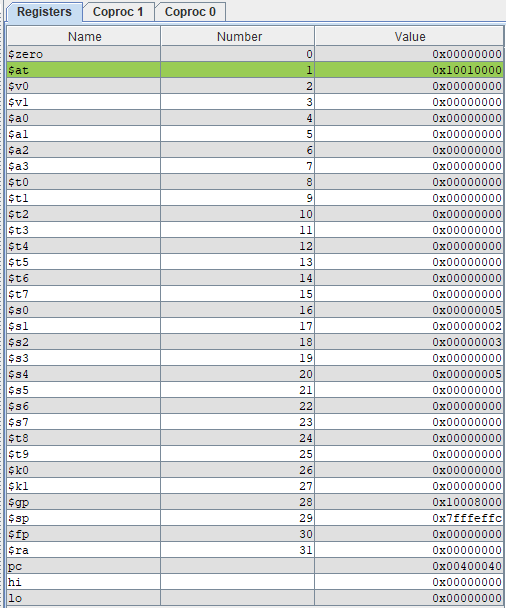
The objective of this test is to demonstrate the knowledge and compare MIPS, Windows 32bit Intel x86 ISA, and Linux 64bit Intel x86-x64 ISA architectures. Examples from Chapter 2 will help show the different concepts of Mips and C language. Other architectures and features will be present that are compared and highlighted throughout this test.

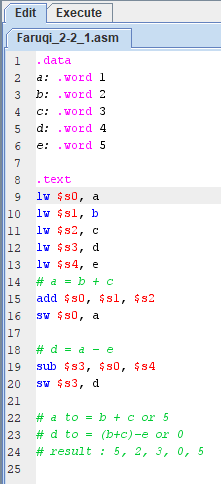
**Part 1 – MIPS**

**2-2\_1.asm**

2-2\_1.asm focuses on showing the relation of registers on basic level. **Figure 1** shows the code written out where a is being assigned the value b + c and d is assigned the value a – e. each static variable is assigned a register and the values are present in the corresponding register. In the end the registers are updated again to hold the new values and the values in memory are also updated since they are all static variables.

**Figure 2** and **Figure 3** show that for each variable a through e have a corresponding register $S0 - $S4 respectively. Before each register can be assigned a value, the address of each variable is loaded from memory to be stored in the $at register, from which the value is retrieved from memory and stored into the correct register.

****For example, the address of static variable a, which is 0x10010000 is first stored in the register $at. Since a is the first value stored in memory location 0x10010000 at offset 0 (+0), the value is retrieved from memory and pushed onto register $S0. The value of a is 0x00000001 in big endian. This is done for the rest of the variables, all pulled from memory since they are static variables in big endian. To perform the operation a = b + c, register $S1 is added to $S2 and stored in $S0. The word is then stored back into memory at offset 0 (+0) in big endian as the value 0x00000005. To perform the operation d = a - e, register $S0 is subtracted from $S4 and stored in $S3. The word is then stored back into memory at offset c (+c) in big endian as the value 0x00000000.



$at stores memory address

$S0 stores value of a

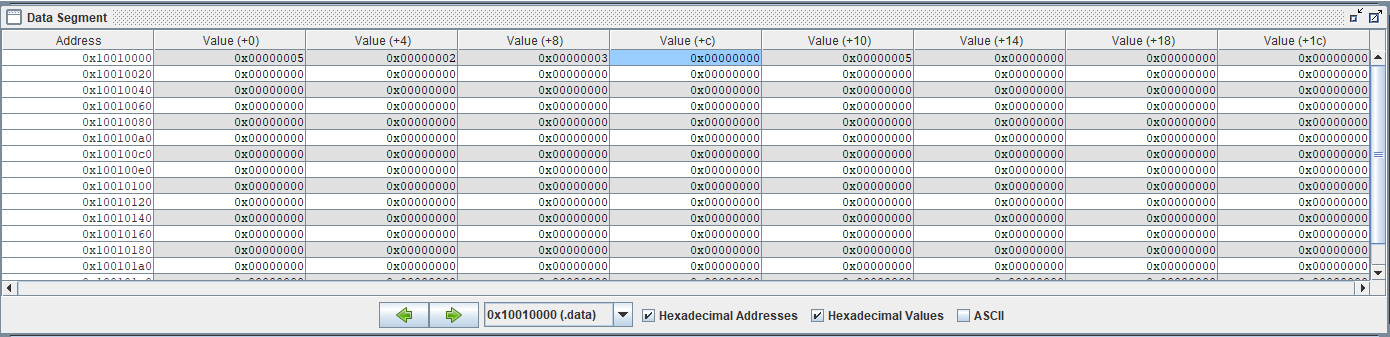
****Variable values are stored in memory and reflect all changes.

Figure 2 - Faruqi\_2-2\_1 MIPS Register

Figure - Faruqi\_2-2\_1 Code

Figure 3 - Faruqi\_2-2\_1 Data Segment

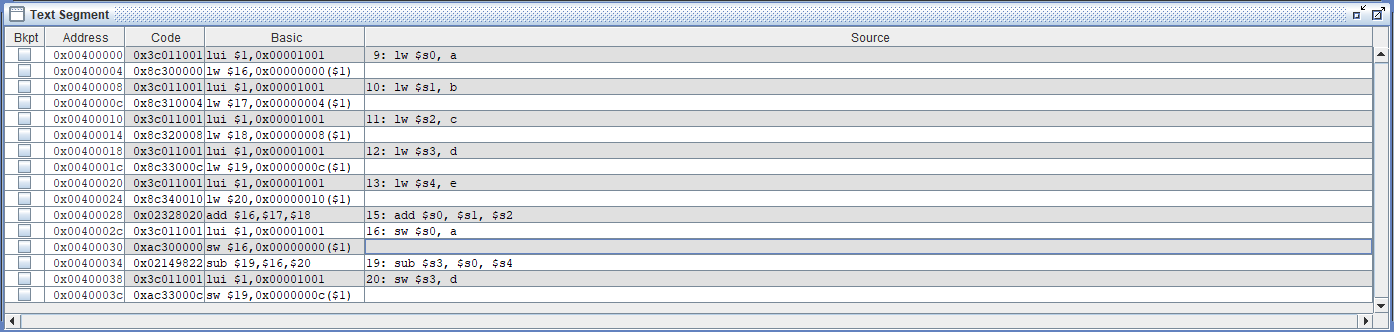
****

Figure 3 - Faruqi\_2-2\_1 Instructions

**2-2\_2.asm**