计算机组成原理实验报告

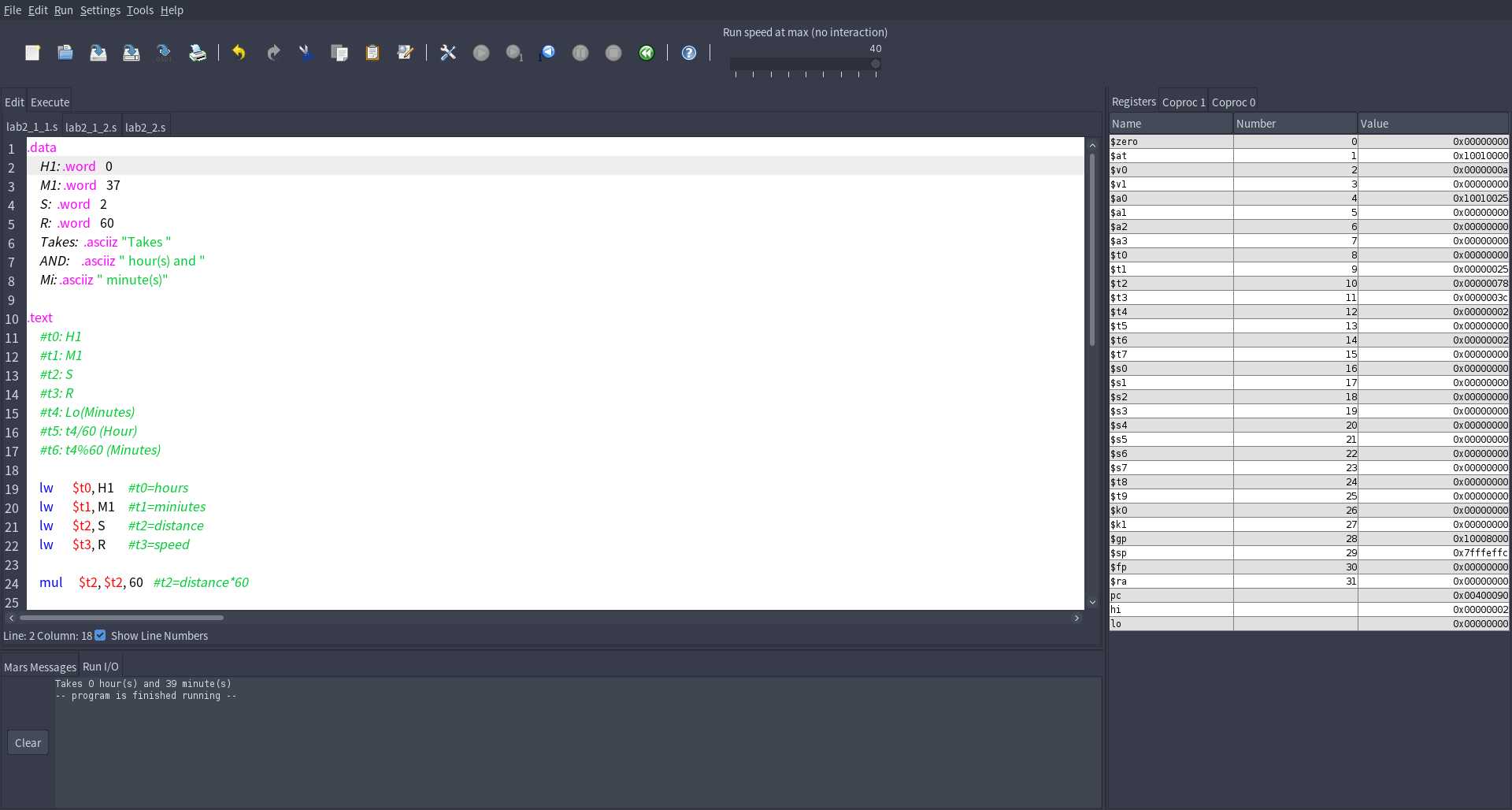
姓名： 吴迪 学号：11810716

1. 实验目的
2. To get familiar with the interface of Mars and QtSpim. Learn how to use them to simulate our coed.
3. Practice using mips instructions. Learn about the registers and memory operations.
4. 实验内容
5. Define data or get data from user input and calculate them, including adding, subtracting, multiplying, dividing. And get a result which is the total time for the student from home to school, in the format that how many hours and minutes.
6. Define a string “abc” in the memory, try to read each character from memory and load them into register. Then do some arithmetic operations to get other ASCII character from the read one, and append it to another string. Finally output the expected string “aAbBcC”.
7. 实验步骤（阐述代码思路或操作步骤）
8. First, the data has two states: one is defined in the file, another is determined by the user. For the first one, we need to define them in the data part. Which are labels for the address to some memory places. For the second one, we can use syscall 5 which is an instruction to read input integer. The input integer will be placed at register $v0. So we get it out and place it in another register or store it to memory.

Then, we have H1, M1, S, R. H1 is the hours from the students home to the transfer station, while M1 is the minutes. S is the distance from the station to school. So we need to calculate the time for the second trip. We can use S/(R/60) because R is kilometers per hours. As R/60 could have remainder, we prefer to multiple S by 60 first. Then we can get the time in minutes. Then minutes divided by 60, we get the hours at register Lo and minutes at Hi. Then we can simply add H1 with Lo and M1 with High. After all, we use syscall 4 to print the result.

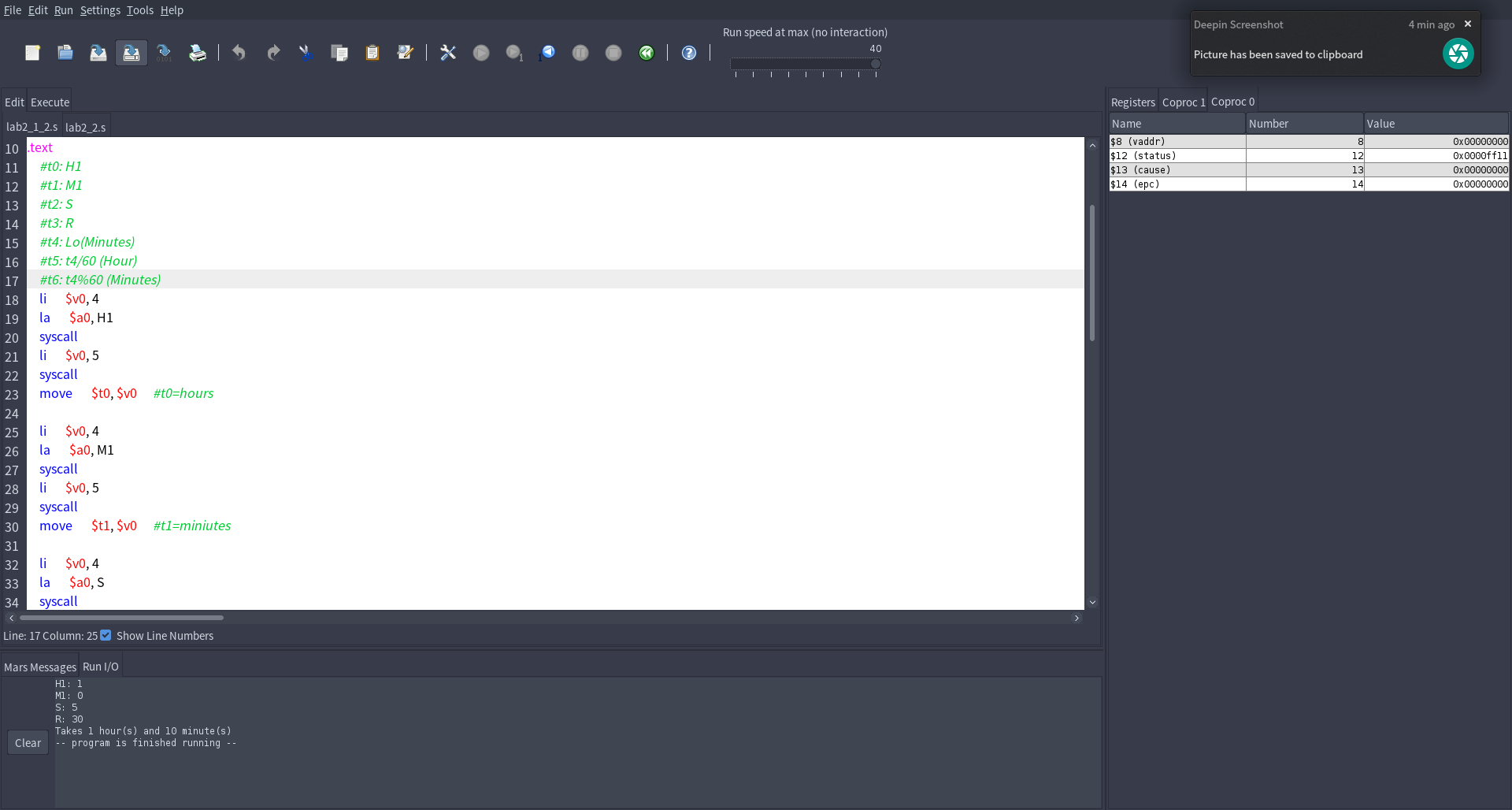
1. First we define “abc” in the data scope and order a place for the result string. Then “abc” will be paced in a memory place. We can use *lb* to get one byte from the string and use *sb* to store the byte we get in the register to the place we preserved. And as the ASCII code rule, to transfer lower case character to upper we can just simply subtract it by 32. Then we just load and store the data one by one with an offset. After all, use syscall 4 to print the result.
2. 实验结果（截图并配以适当的文字说明）

1\_1



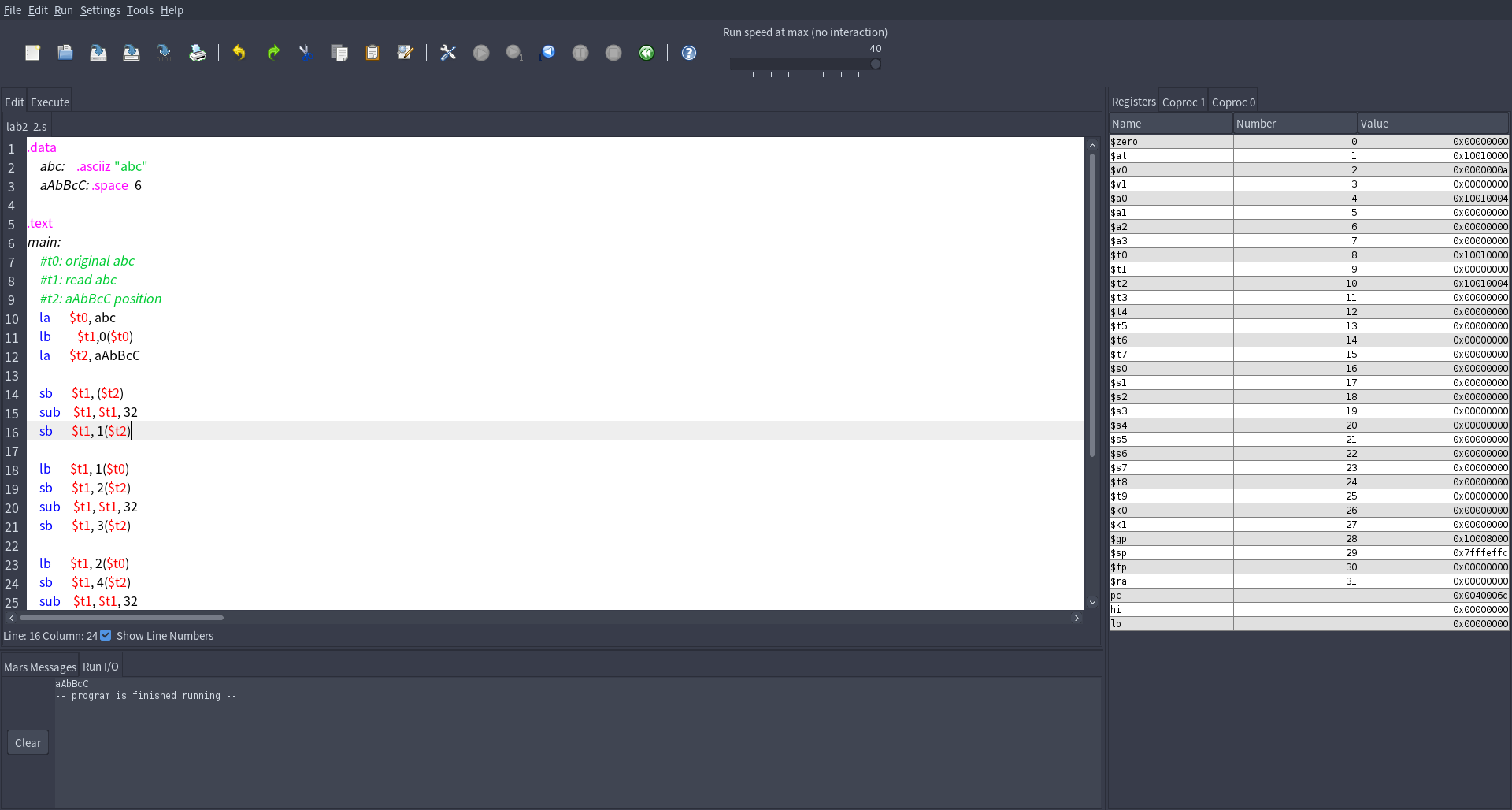
We can see that the defined H1 is 0 hour and M1 is 37 minutes, S is 2 kilometers and R is 60 kilometers per hours. The output is 0 hour and 39 minutes, which matches the excepted.

1\_2



We can see in the console that we input H1 to be 1 and M1 to be 0. S is 5 kilometers and R is 30 kilometers per hours. The output is 1 hour and 10 minutes, which matched our exception.

2



We can see we defined a string called abc and store “abc”. The output is “aAbBcC” which matches our exception.

1. 实验分析（遇到的问题以及解决方案）
2. For the data we defined in the data scope, actually they are just label for the address in the memory. So if we want to load the values, we need to use *la* to store it to a register and use brackets “()” to get the value from the memory correspond to the address.
3. For the usage of *mult* and *div*, they both use special registers Hi and Lo. For *mult*, Hi store the overflow of multiplication, Lo store the normal bytes. And for *div*, Hi store the remainder and Lo store the integer result or division. Don’t know why, at first I kept thinking Hi store the result for *div*. May be is because the position?
4. 实验小结与体会

The instructions of mips are simple. But if we use it to implement a complicated task, it may be hard to compose them to work. We need to use a lot of different instructions to do even addition.

The transfer between register and memory is sometimes annoying. It’s said that the data in registers are faster to get. But we need to keep refreshing the data in the registers, and the amount is limited. So we must to store the necessary data to memory.