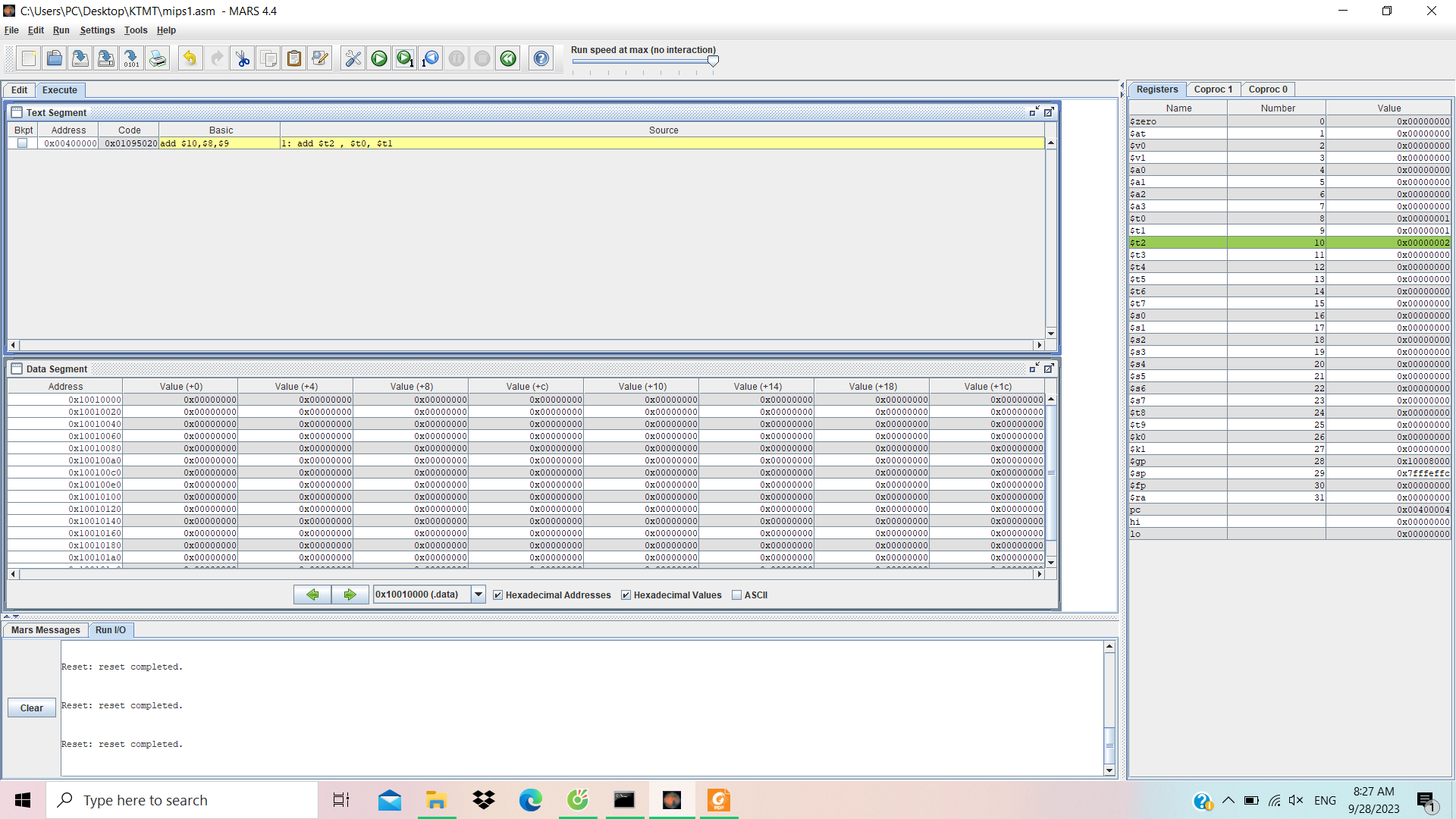
**BẢN BÁO CÁO**

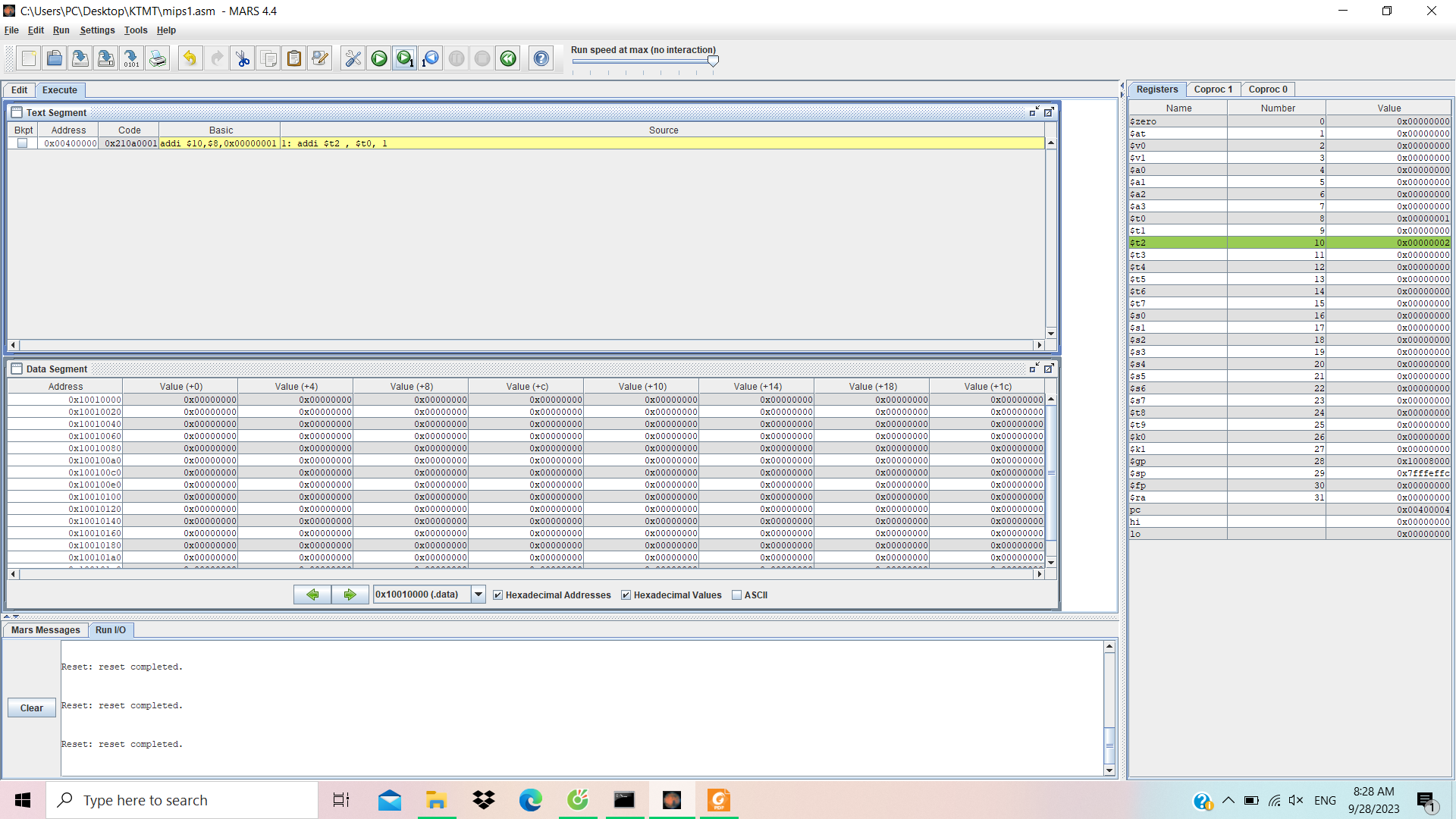
**2. Thực hành**

**2.1**

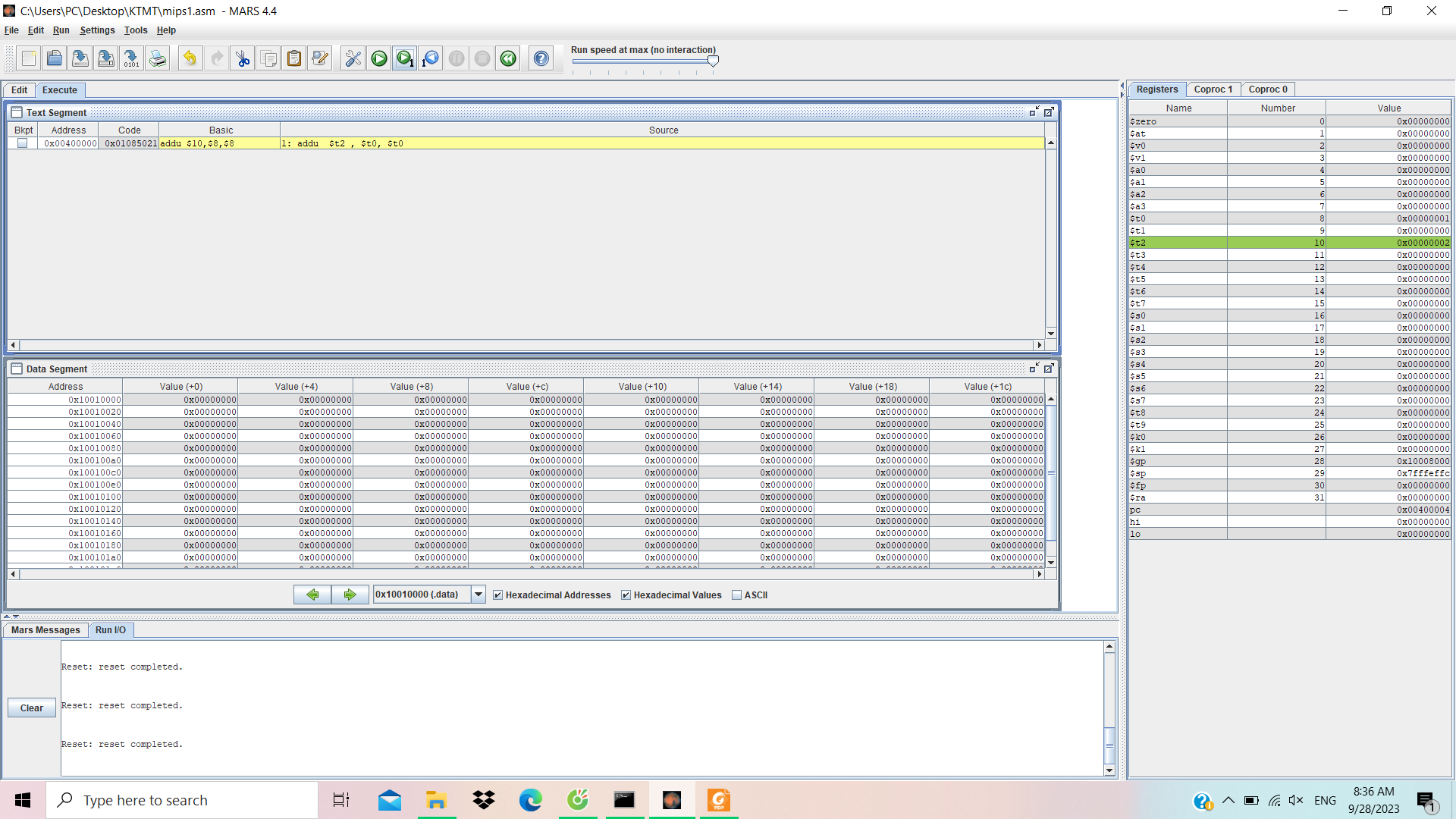
Lệnh add có chức năng cộng giá trị của hai thanh ghi và lưu kết quả vào một thanh ghi khác



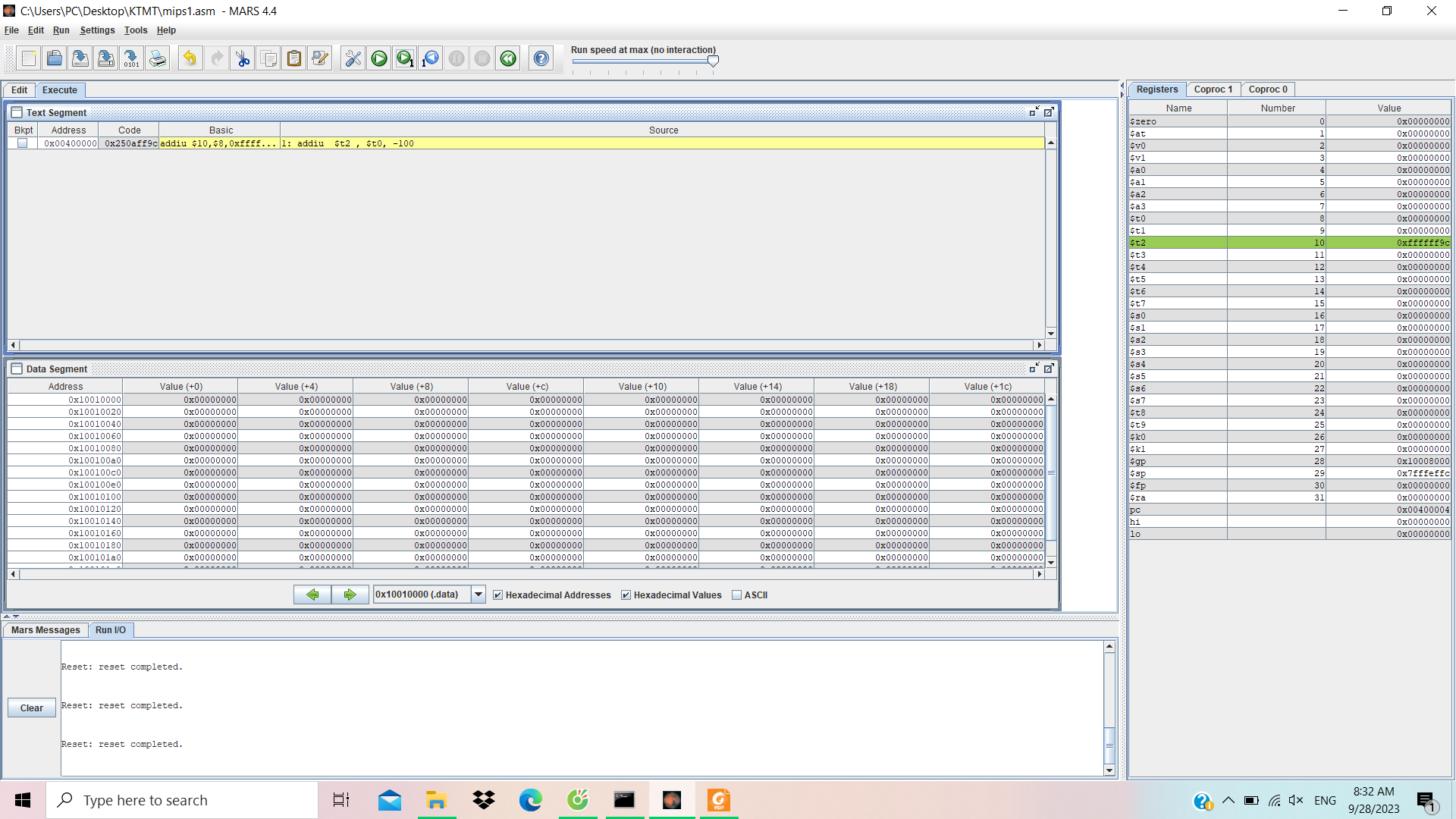
Lệnh addi có chức năng cộng giá trị của thanh ghi với một số bất kỳ và lưu kết quả vào một thanh ghi khác



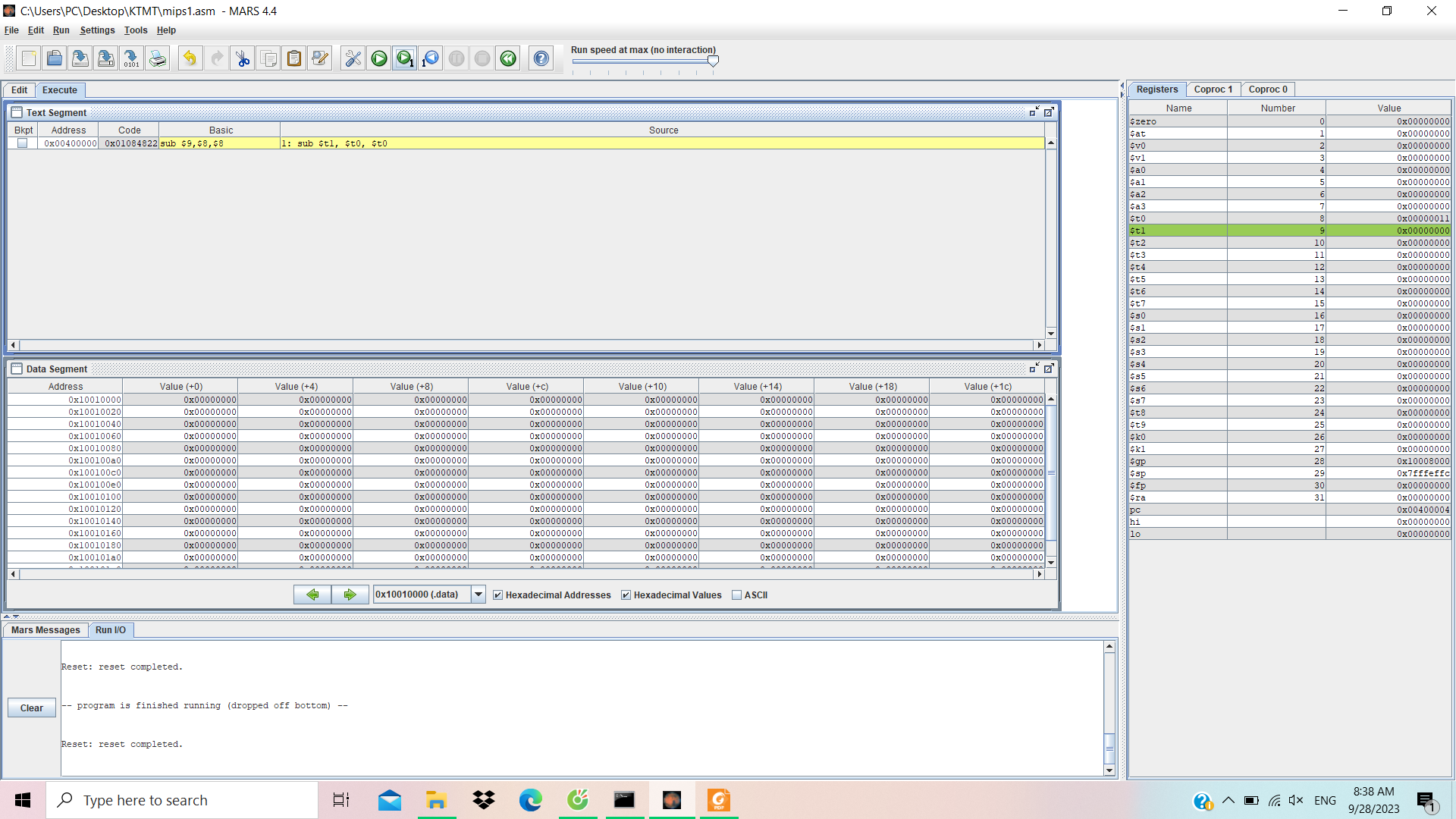
Lệnh addu có chức năng giống lệnh add nhưng chỉ khác là không có cảnh báo tràn



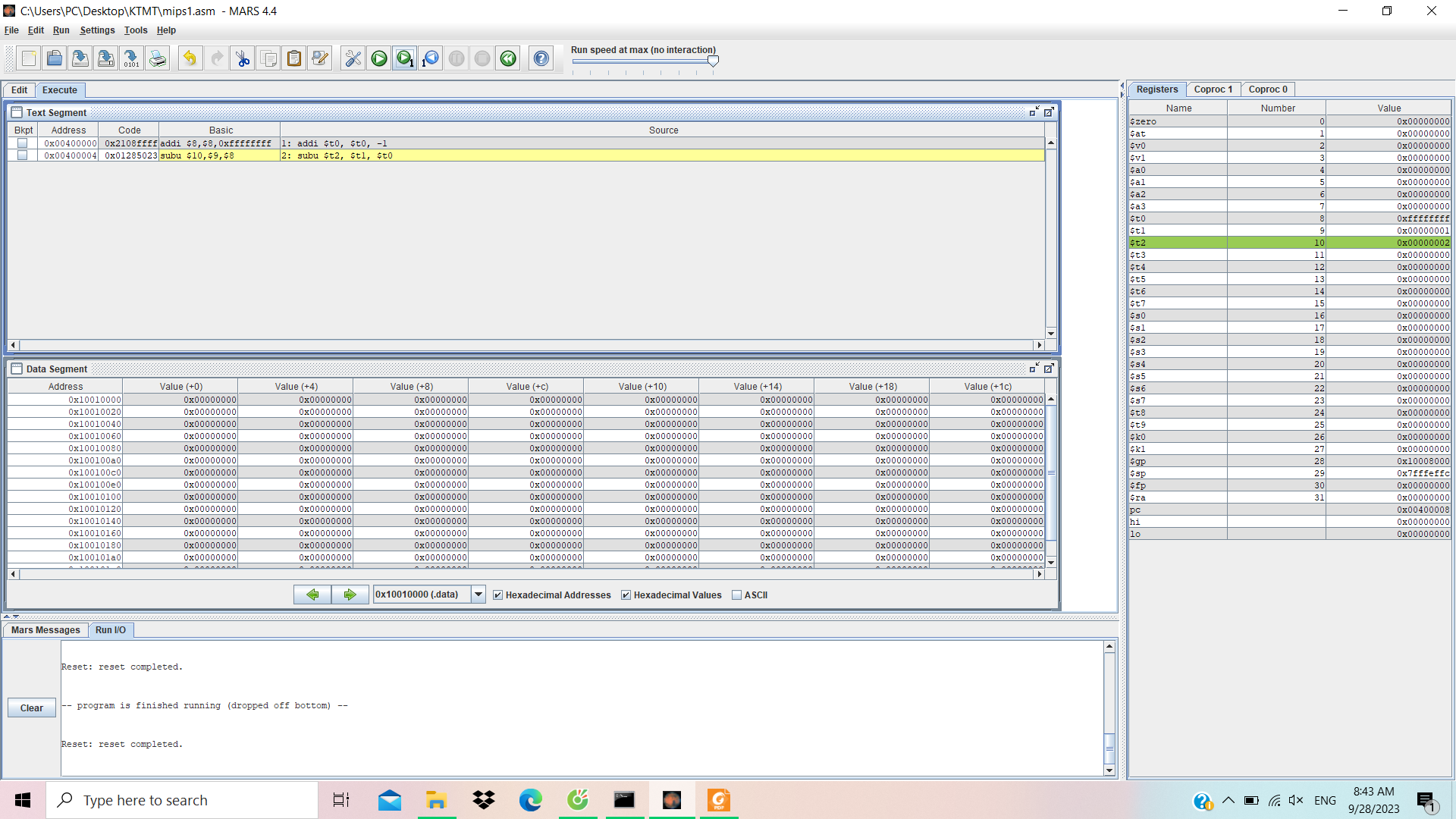
Lệnh addiu có chức năng giống addi nhưng không có cảnh báo tràn



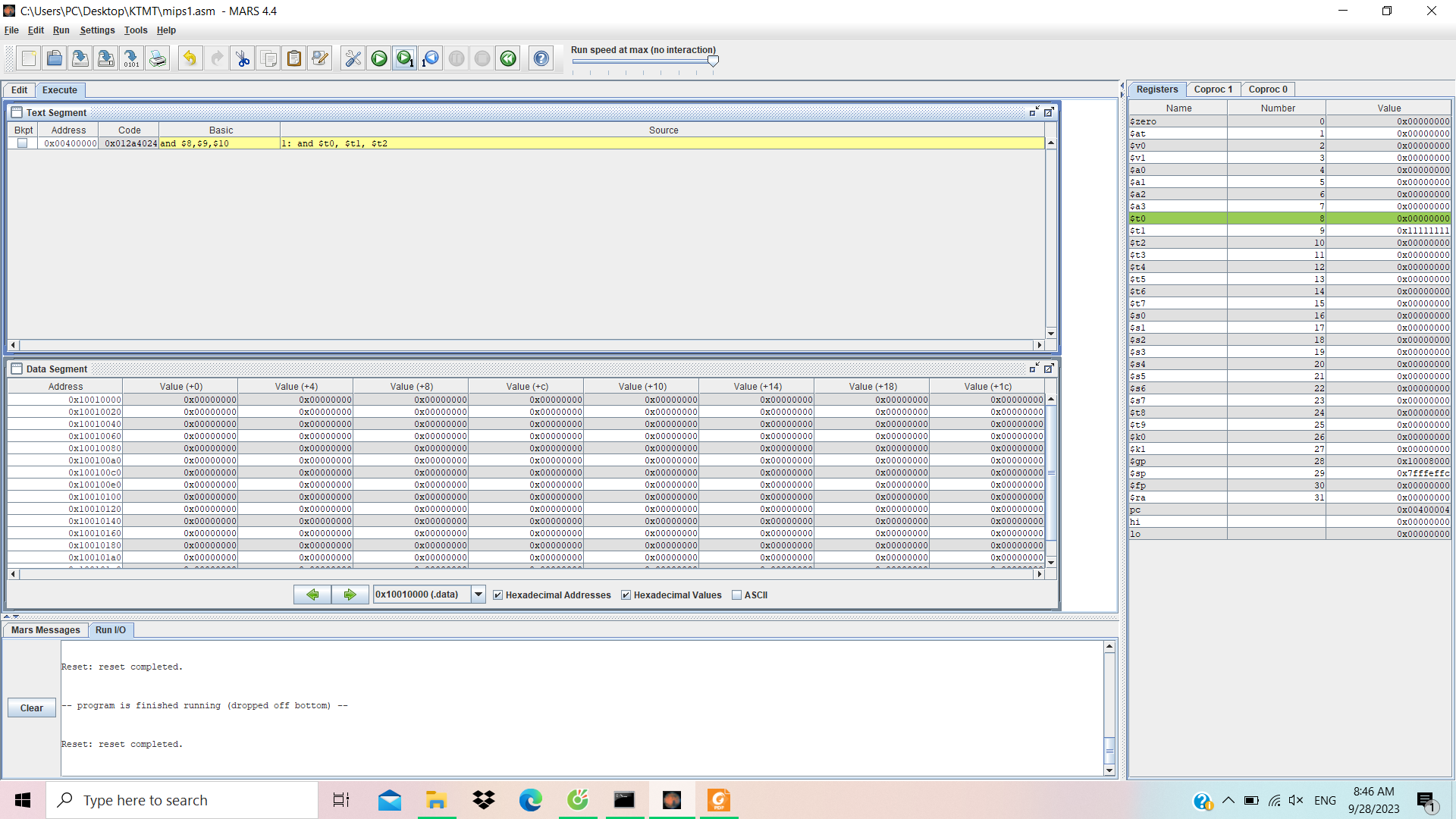
Lệnh sub có chức năng trừ giá trị của hai thanh ghi với nhau và lưu kết quả vào thanh ghi thứ 3



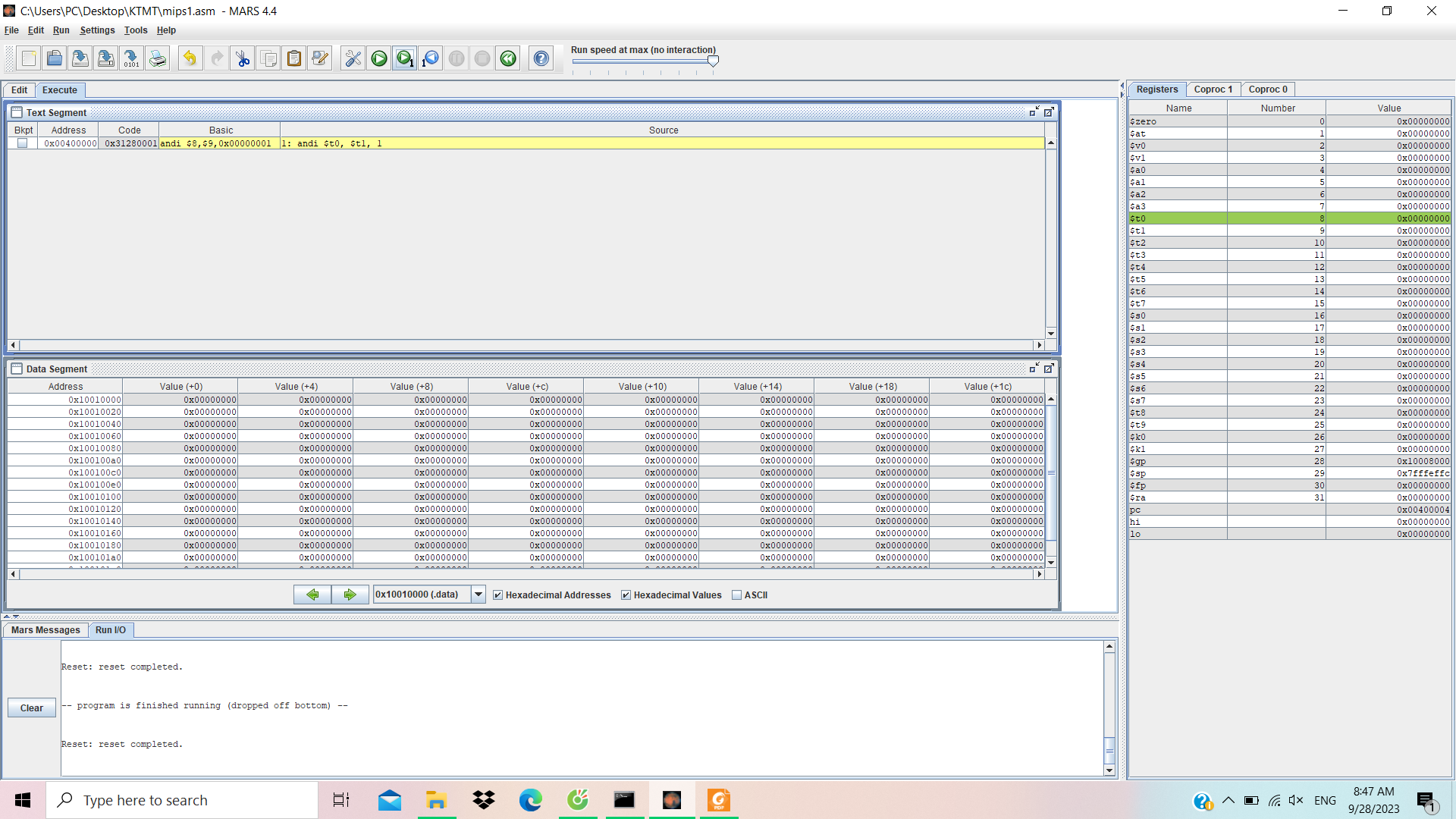
Lệnh subu có chức năng giống lệnh sub nhưng không có cảnh báo tràn



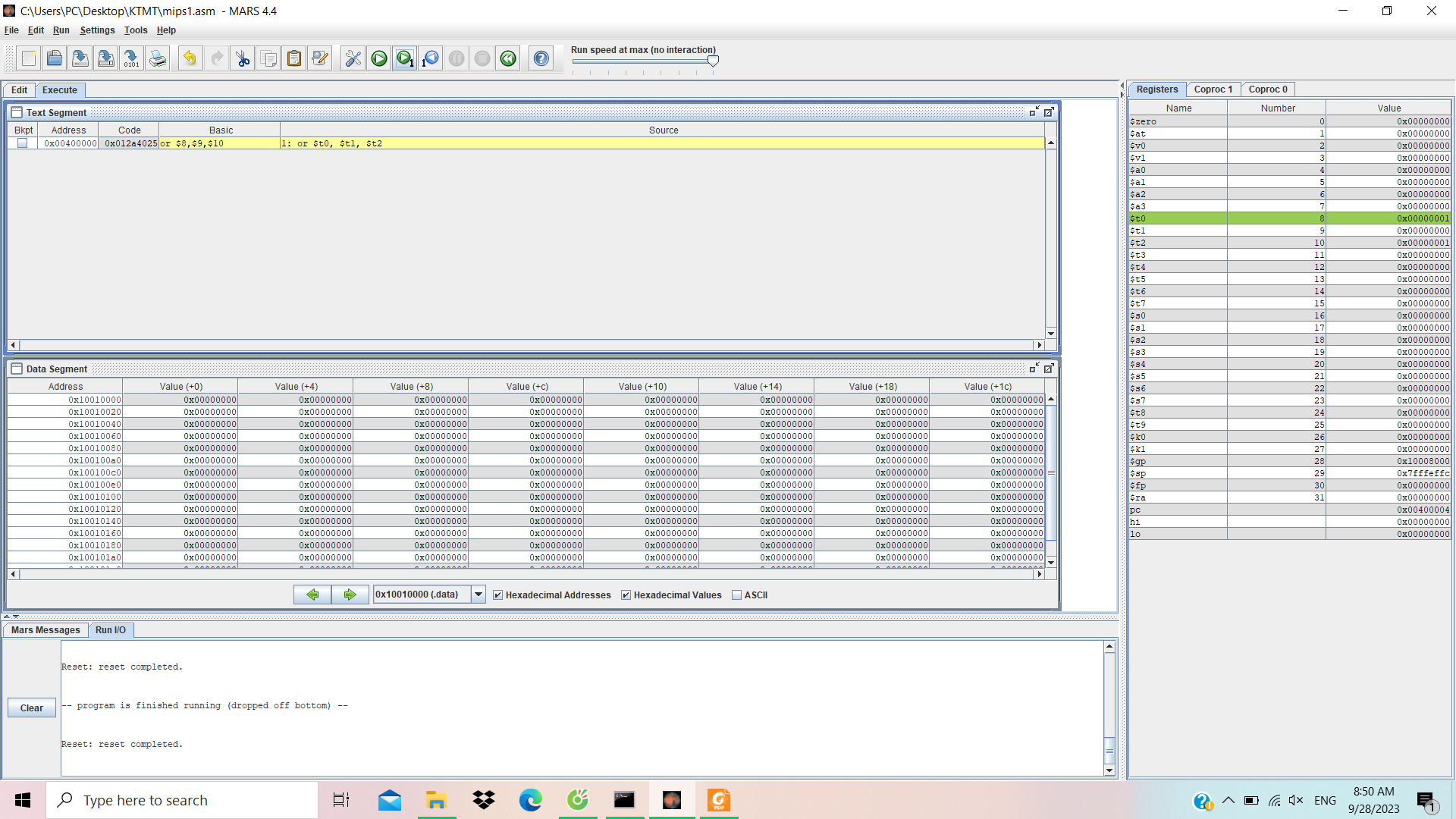
Lệnh and có chức năng so sánh từng bit của giá trị thuộc hai thanh ghi , cho kết quả theo logic của cổng and và lưu vào thanh ghi thứ ba



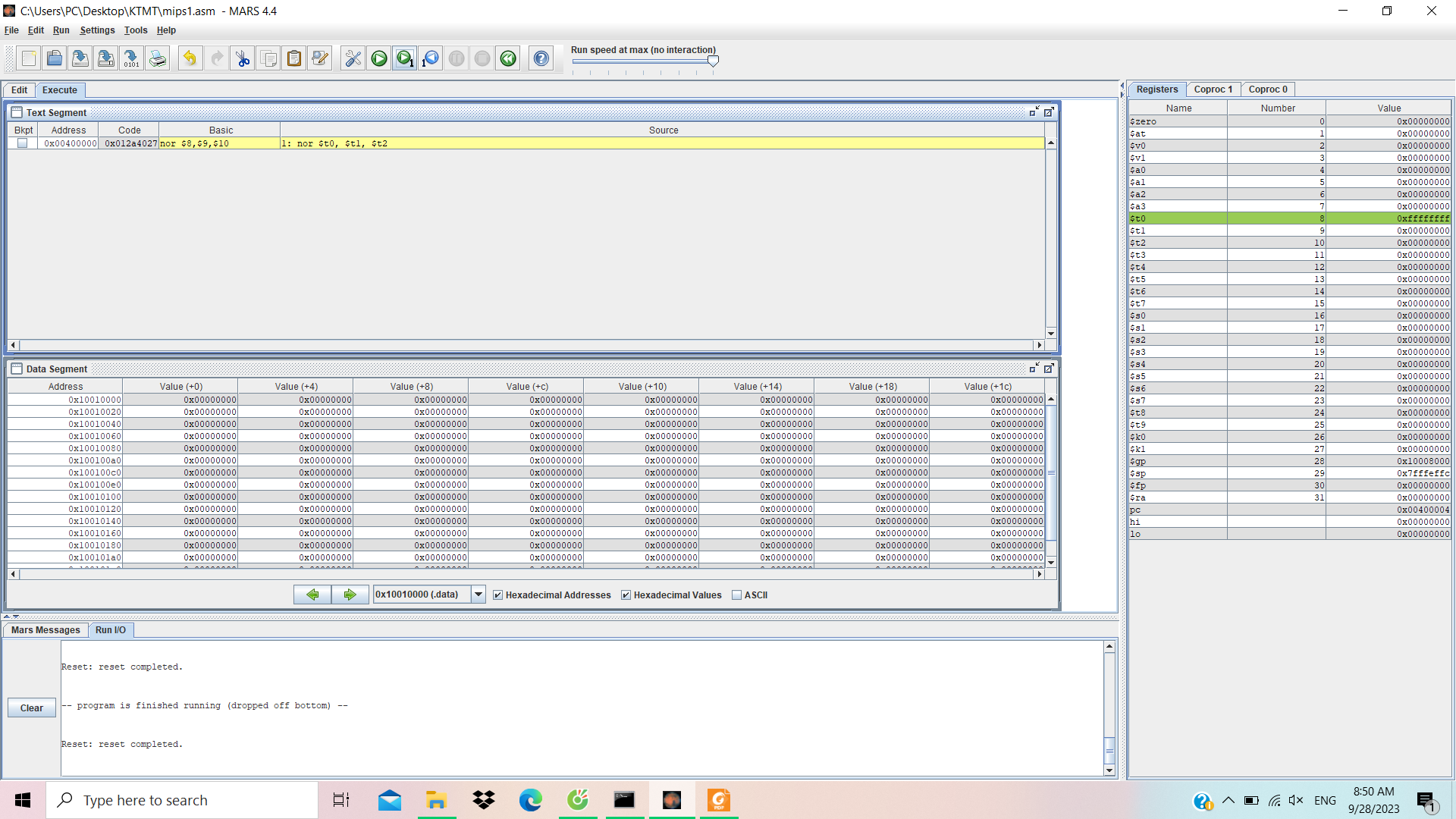
Lệnh andi có chức năng so sánh từng bit của giá trị thanh ghi với từng bit giá trị của một số bất kỳ (dưới dạng nhị phân), cho kết quả theo logic cổng and và lưu vào thanh ghi thứ hai



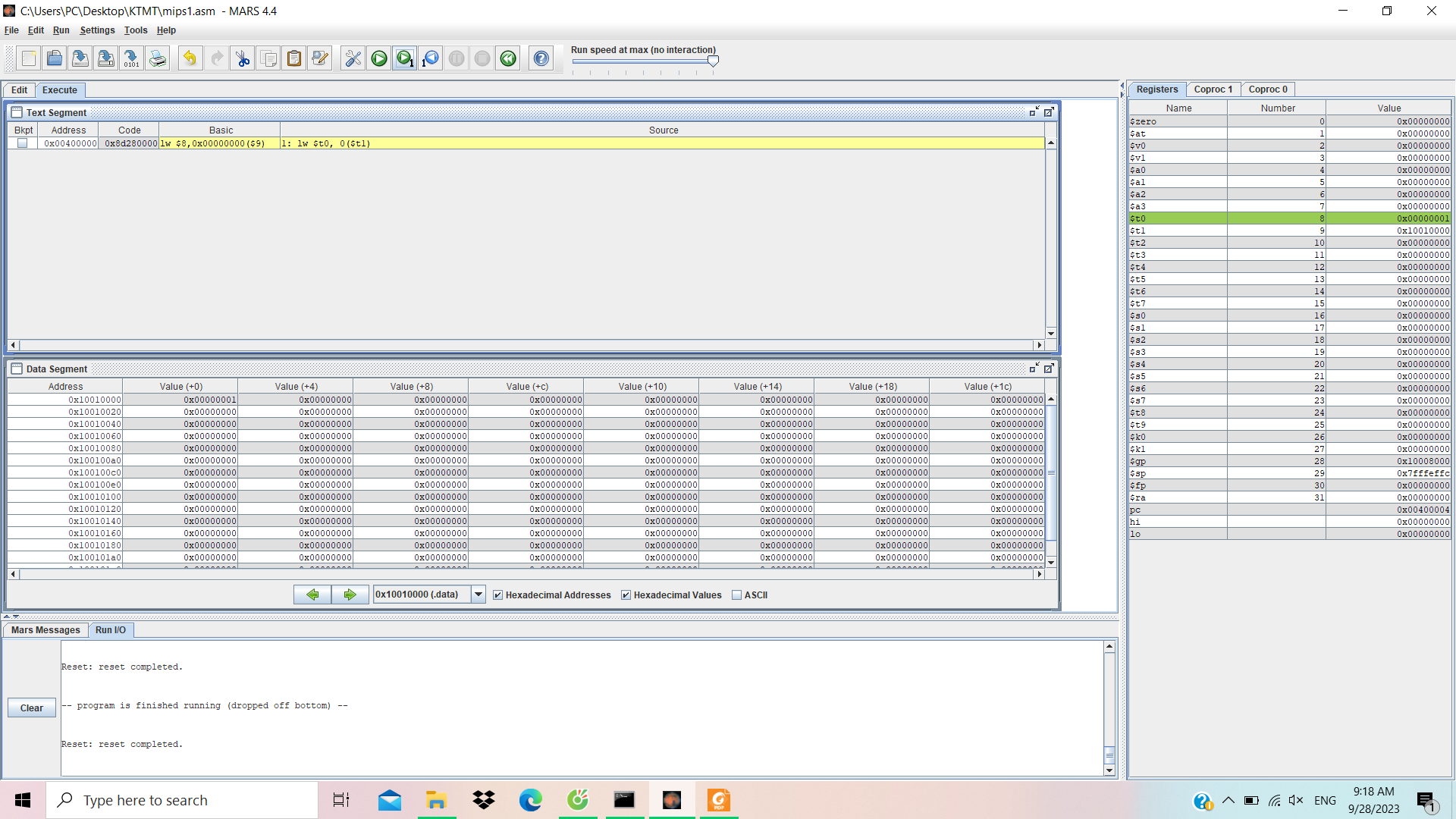
Lệnh or có chức năng so sánh từng bit của giá trị thuộc hai thanh ghi, cho kết quả theo logic cổng or và lưu vào thanh ghi thứ ba



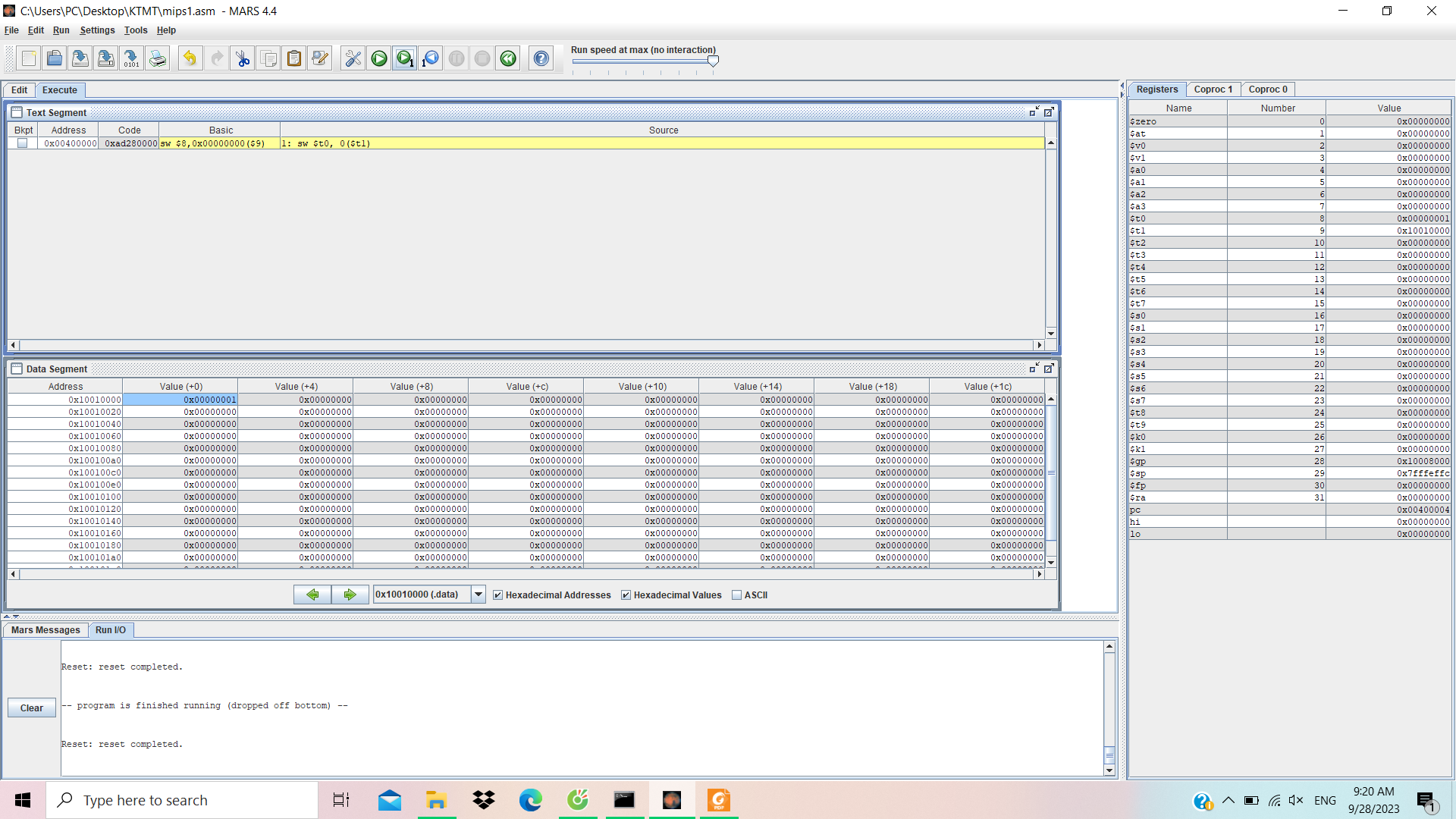
Lệnh nor có chức năng so sánh từng bit của giá trị thuộc hai thanh ghi với nhau, cho kết quả theo logic cổng nor và lưu vào thanh ghi thứ ba



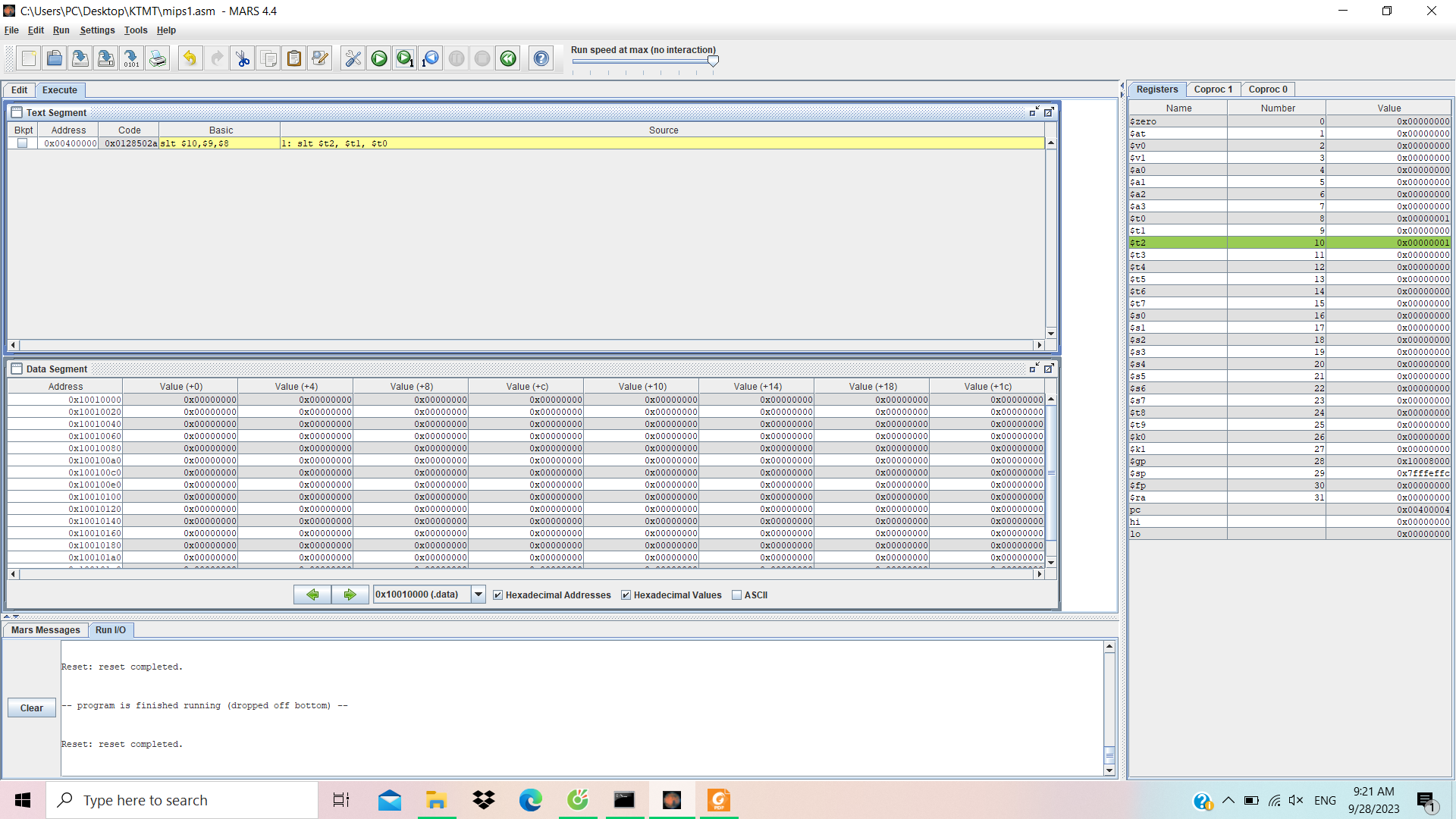
Lệnh lw có chức năng lấy giá trị từ một địa chỉ của thanh ghi và đưa vào một thanh ghi khác



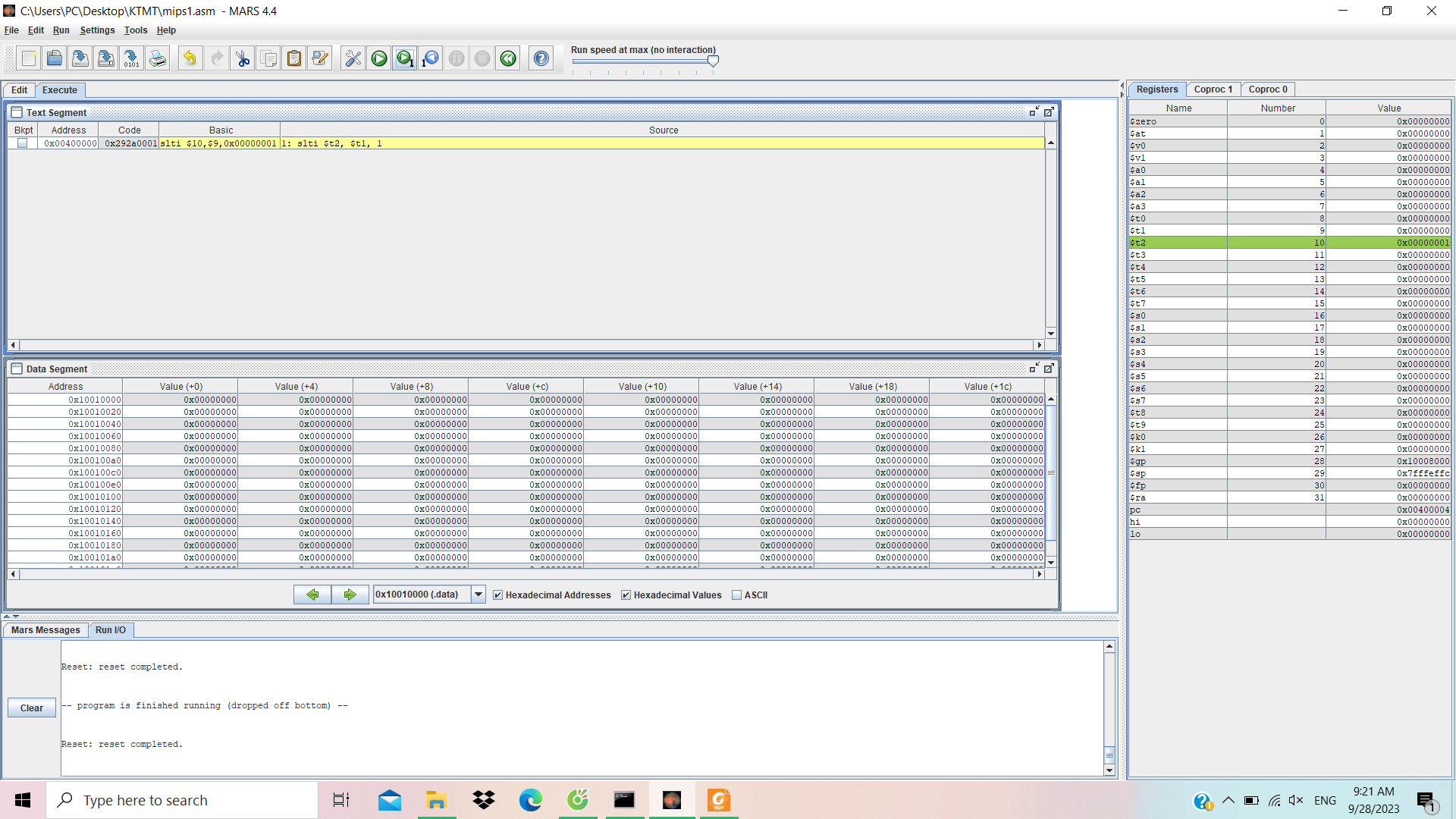
Lệnh sw có chức năng lấy giá trị của một thanh ghi lưu xuống một địa chỉ của thanh ghi khác



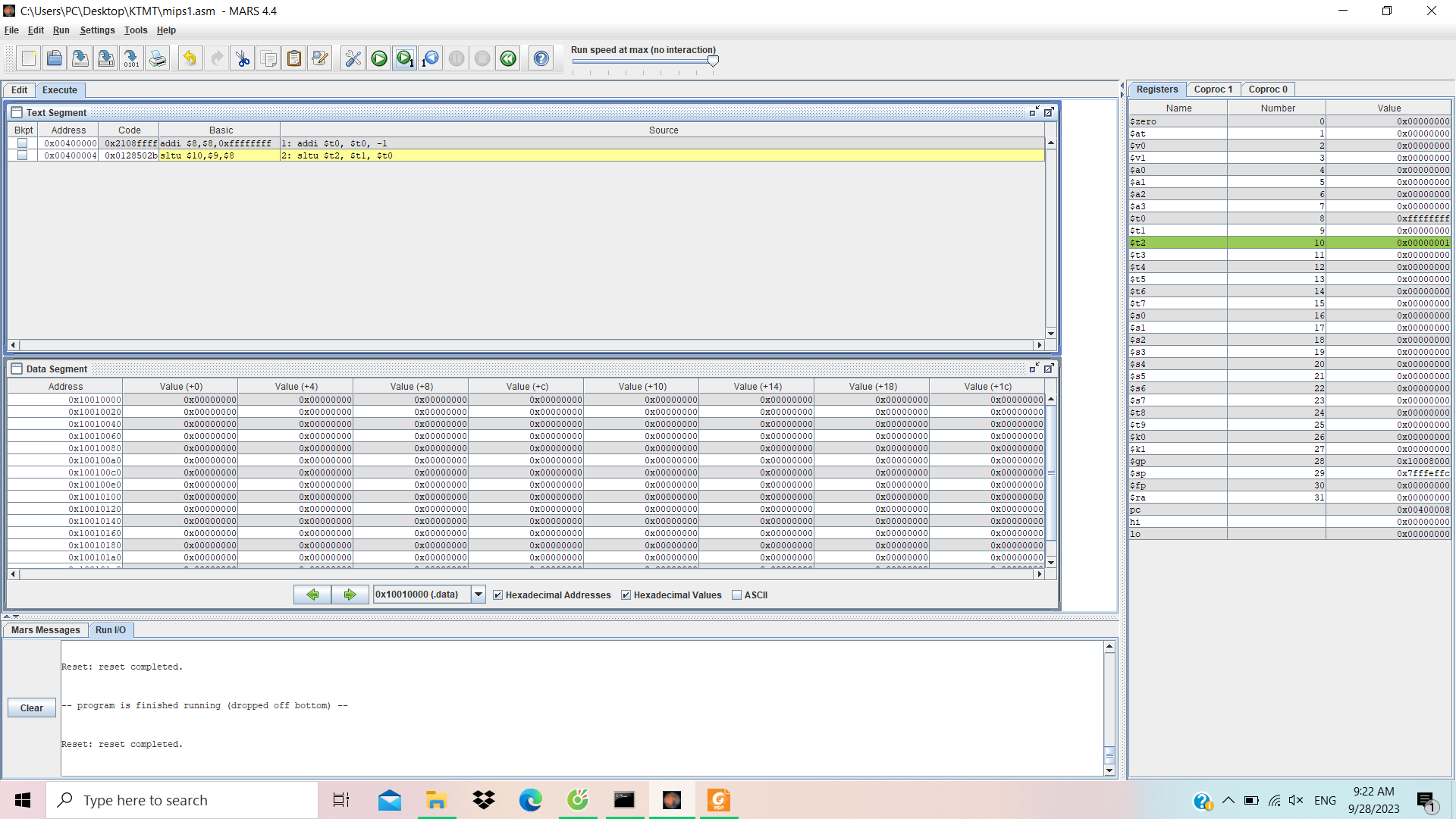
Lệnh slt có chức năng so sánh giá trị của thanh ghi A có nhỏ hơn giá trị của thanh ghi B hay không, nếu có thì cho giá trị thanh ghi lưu kết quả bằng 1. Ngược lại thì cho bằng 0



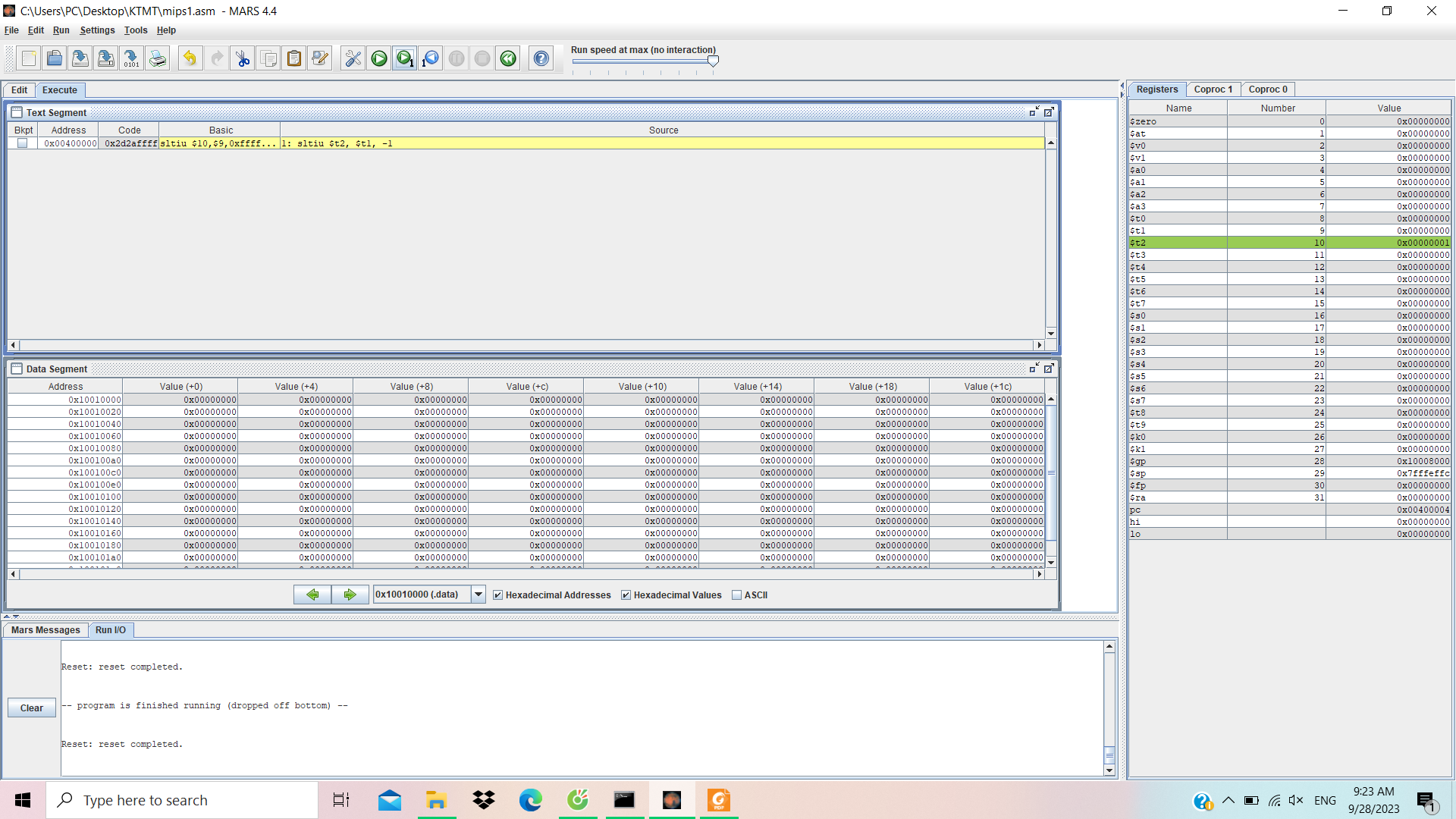
Lệnh slti có chức năng giống với lệnh slt chỉ khác là lệnh này so sánh giá trị của một thanh ghi với một số



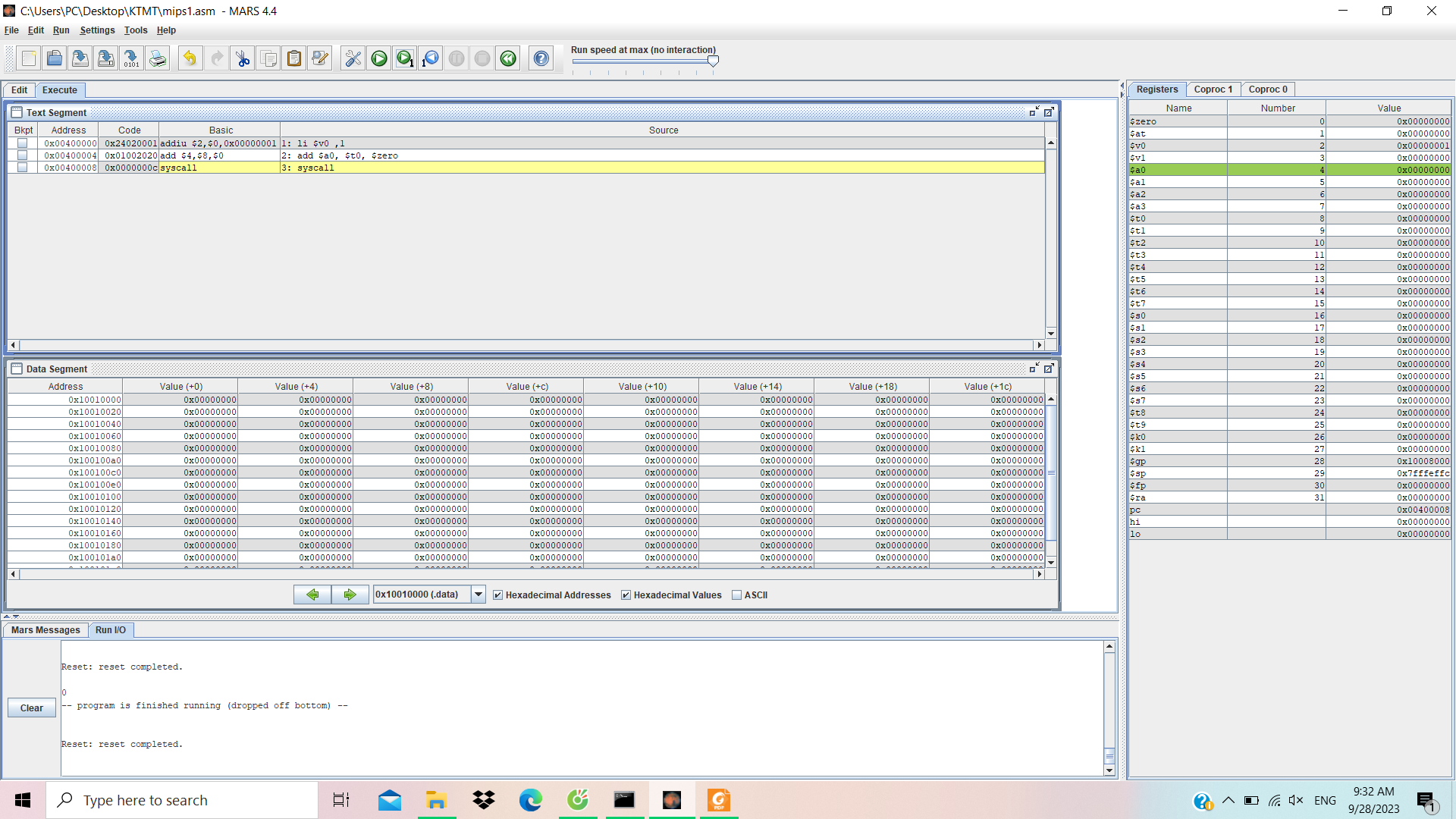
Lệnh sltu có chức năng slt nhưng giá trị của hai thanh ghi so sánh là không có dấu



Lệnh sltiu có chức năng giống lệnh slti nhưng giá trị của thanh ghi và số so sánh là không có dấu

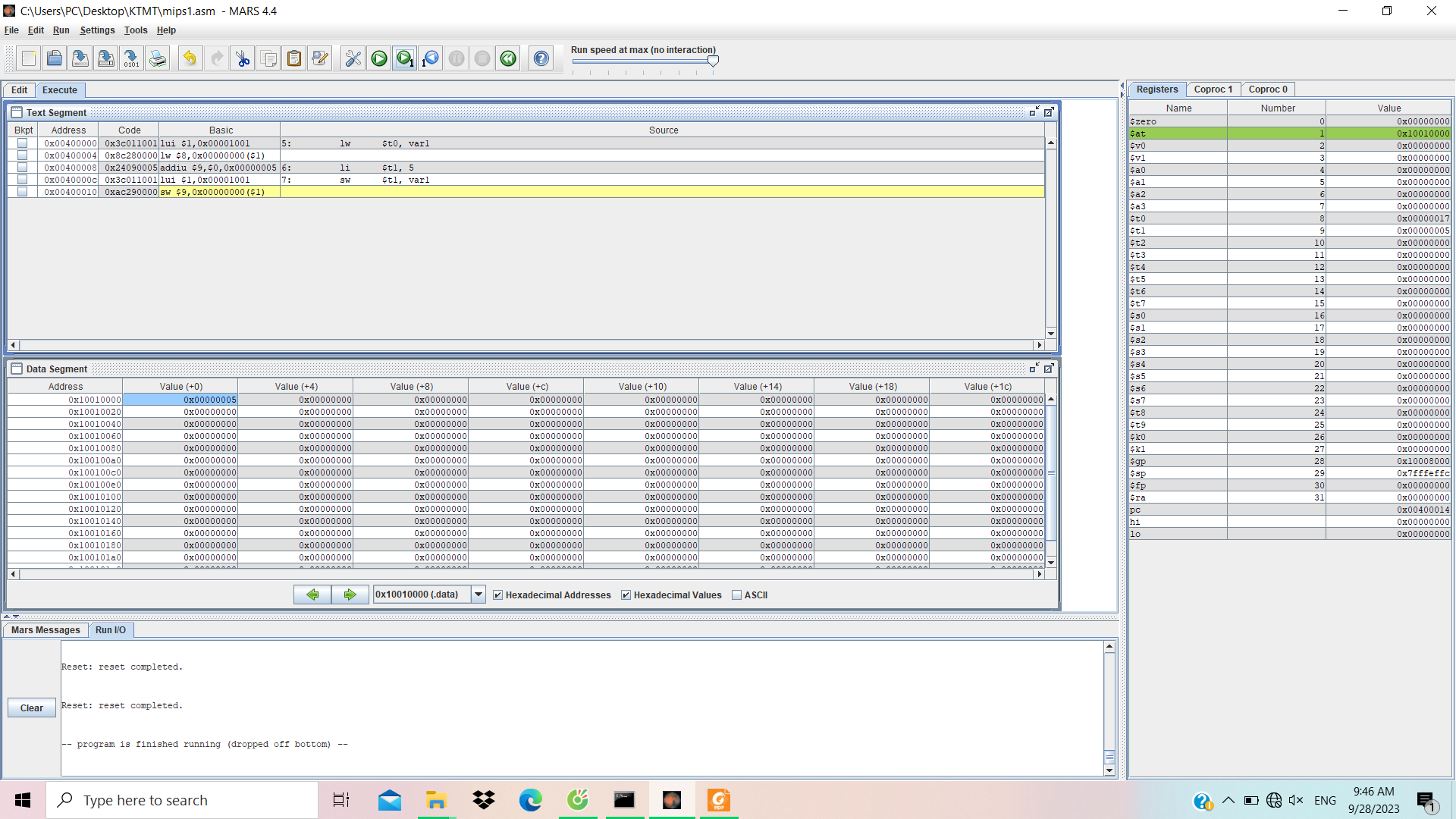


Lệnh syscall có chức năng phụ thuộc theo giá trị của thanh ghi $v0, vd sau đây $v0 có giá trị là 1 đồng nghĩa với syscall có chức năng in số nguyên được lưu trong thanh ghi $a0 ra màn hình



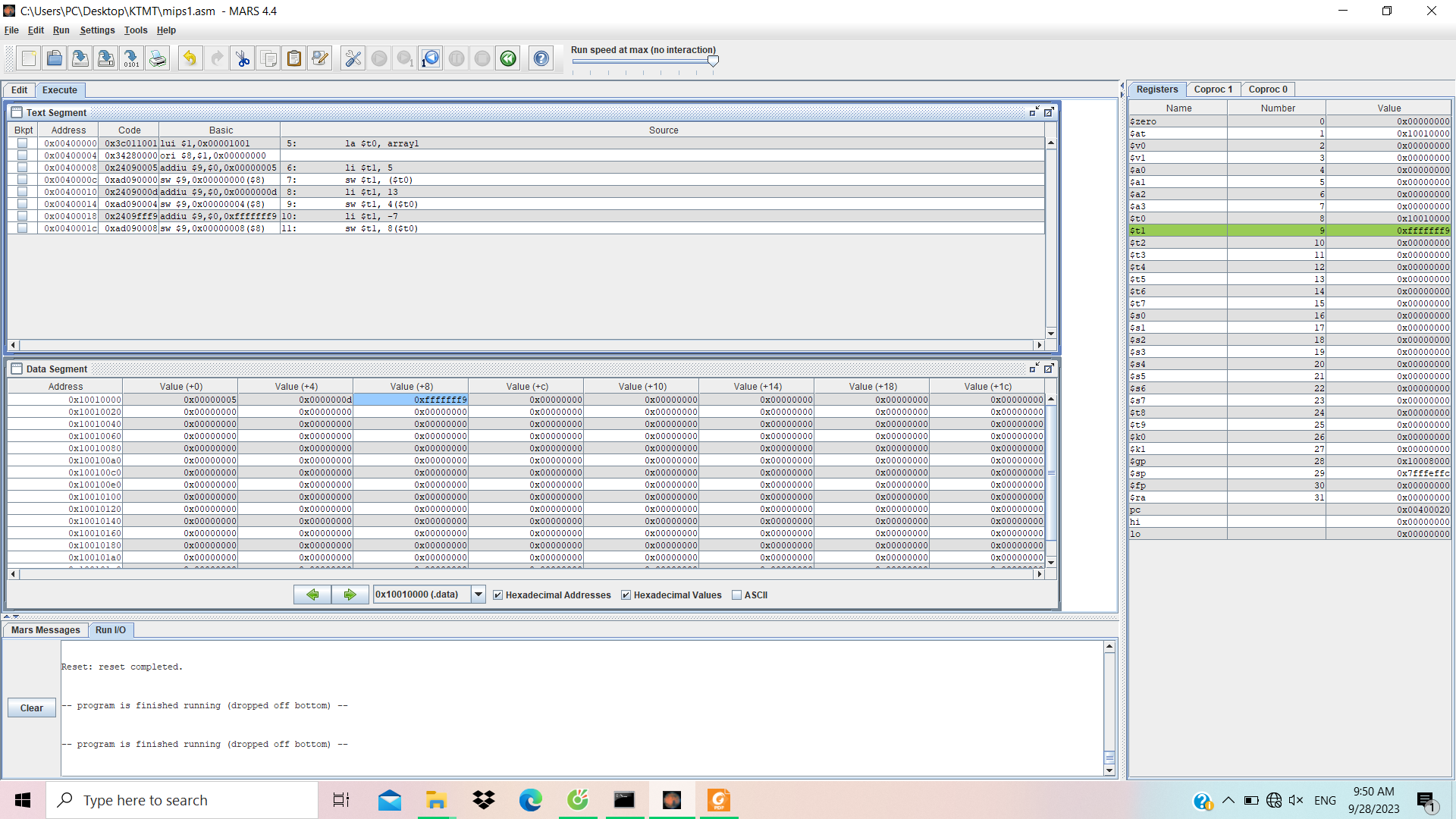
**2.2**

VD1:

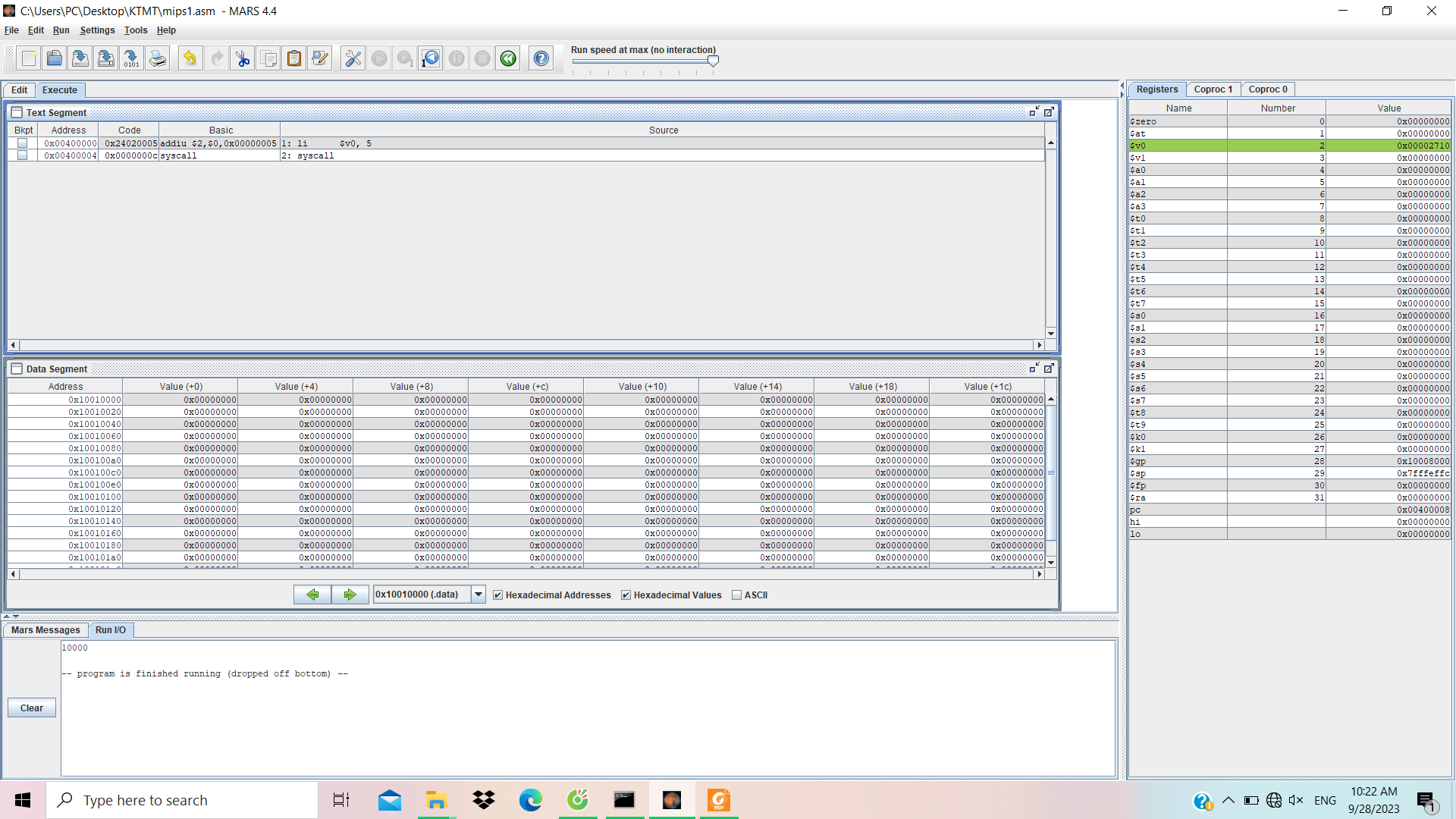


Ý nghĩa: chương trình khai báo var1 với giá trị ban đầu là 17 (hệ thập lục phân), sau đó lưu giá trị của var1 vào thanh ghi $t0 và thay đổi giá trị của var1 sang 5

VD2:

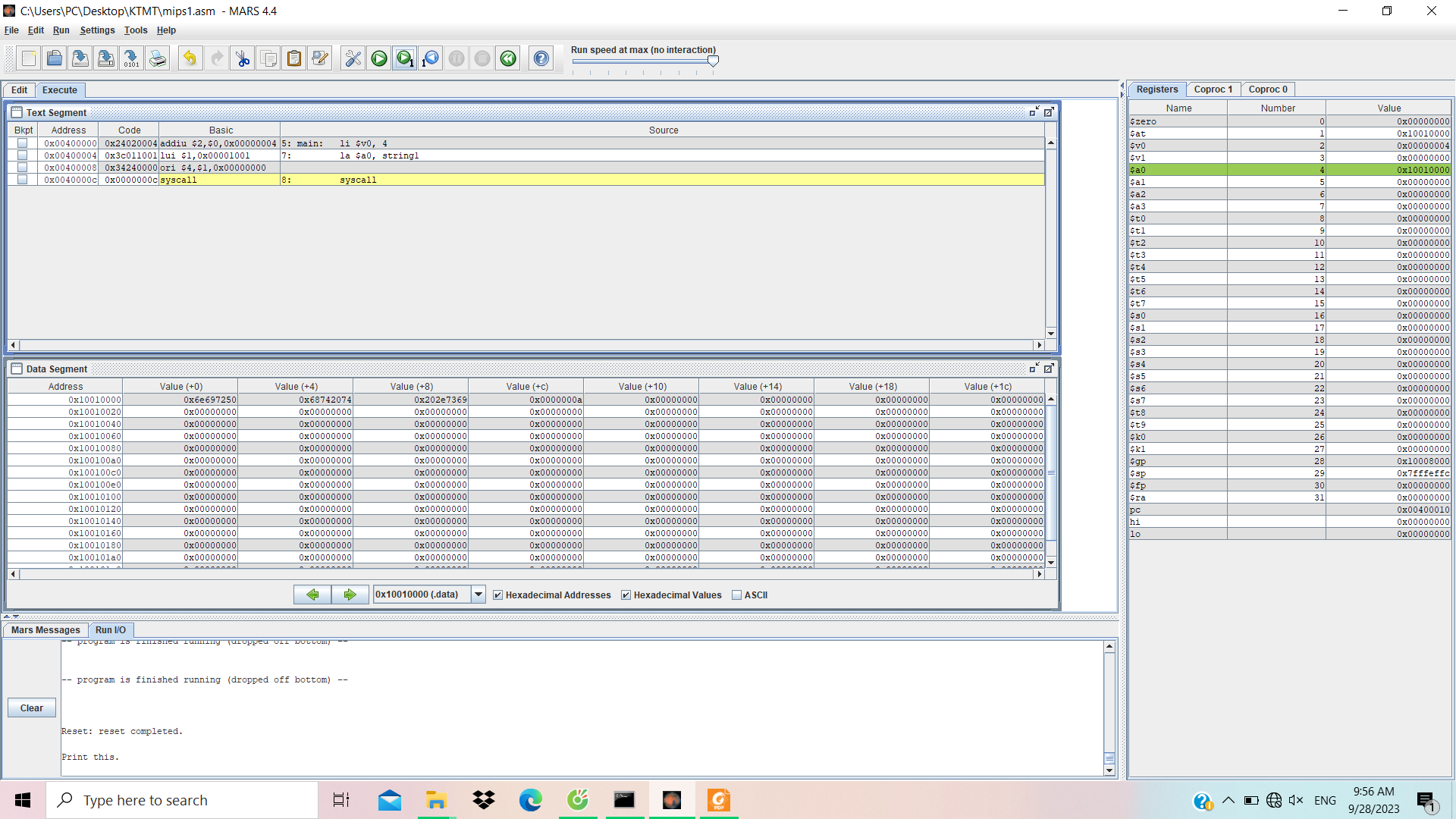


Ý nghĩa: chương trình khởi tạo biến array1 với 12 ô địa chỉ, sau đó chương trình lấy địa chỉ của array1 sau đó thay đổi giá trị ở địa chỉ 0x10010000 của array1 thành 516. Tiếp đến chương trình thay đổi giá trị ở địa chỉ 0x10010004 sang d16. Sau cùng thay đổi giá trị ở ô nhớ 0x10010008 thành -710

VD3:  


Ý nghĩa: chương trình gán giá trị 5 cho $v0 sau đó gọi syscall. Syscall sẽ đọc $v0 và nhận thấy giá trị 5 tương ứng với chức năng lưu số từ input vào $v0

VD4:

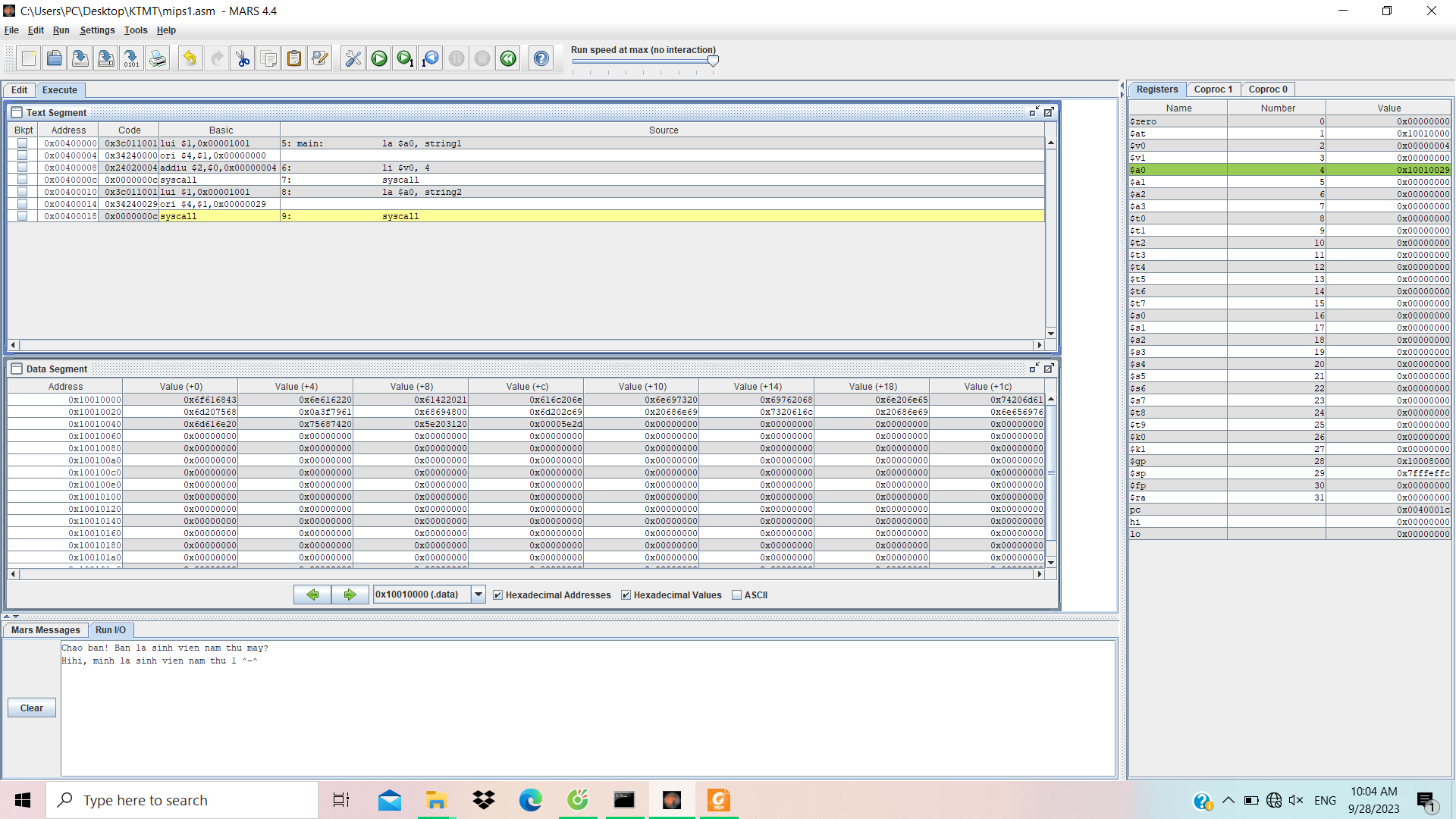
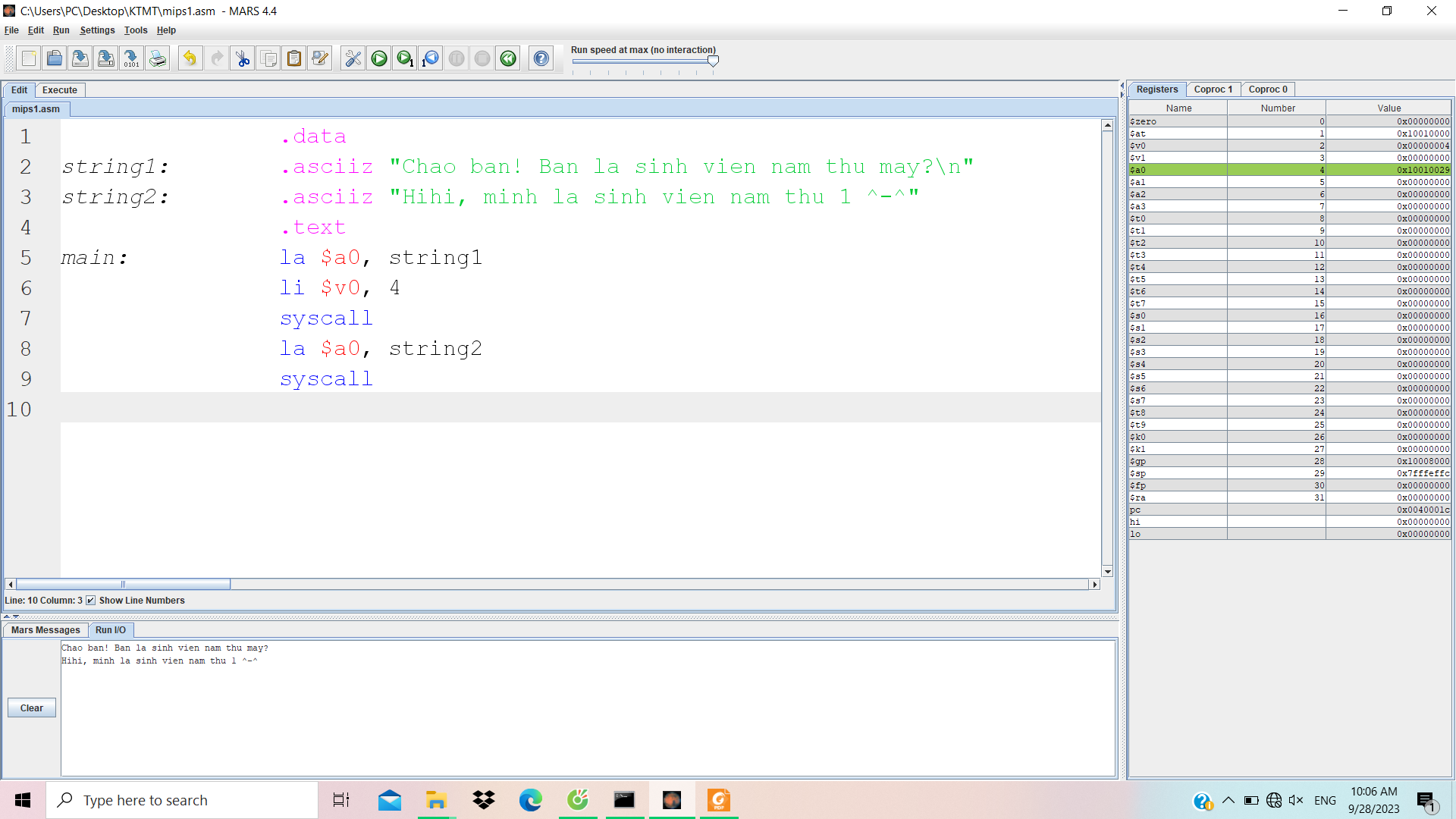


Ý nghĩa: chương trình khai báo biến string1, sau đó gán giá trị 4 cho $v0 và gán địa chỉ đầu tiên của string1 vào thanh ghi $a0 rồi gọi syscall. Syscall đọc giá trị được giá trị $v0 là 4 và thực hiện chức năng in chuỗi thuộc địa chỉ mà $a0 lưu

**3 Bài tập**

**3.1**

**a)**

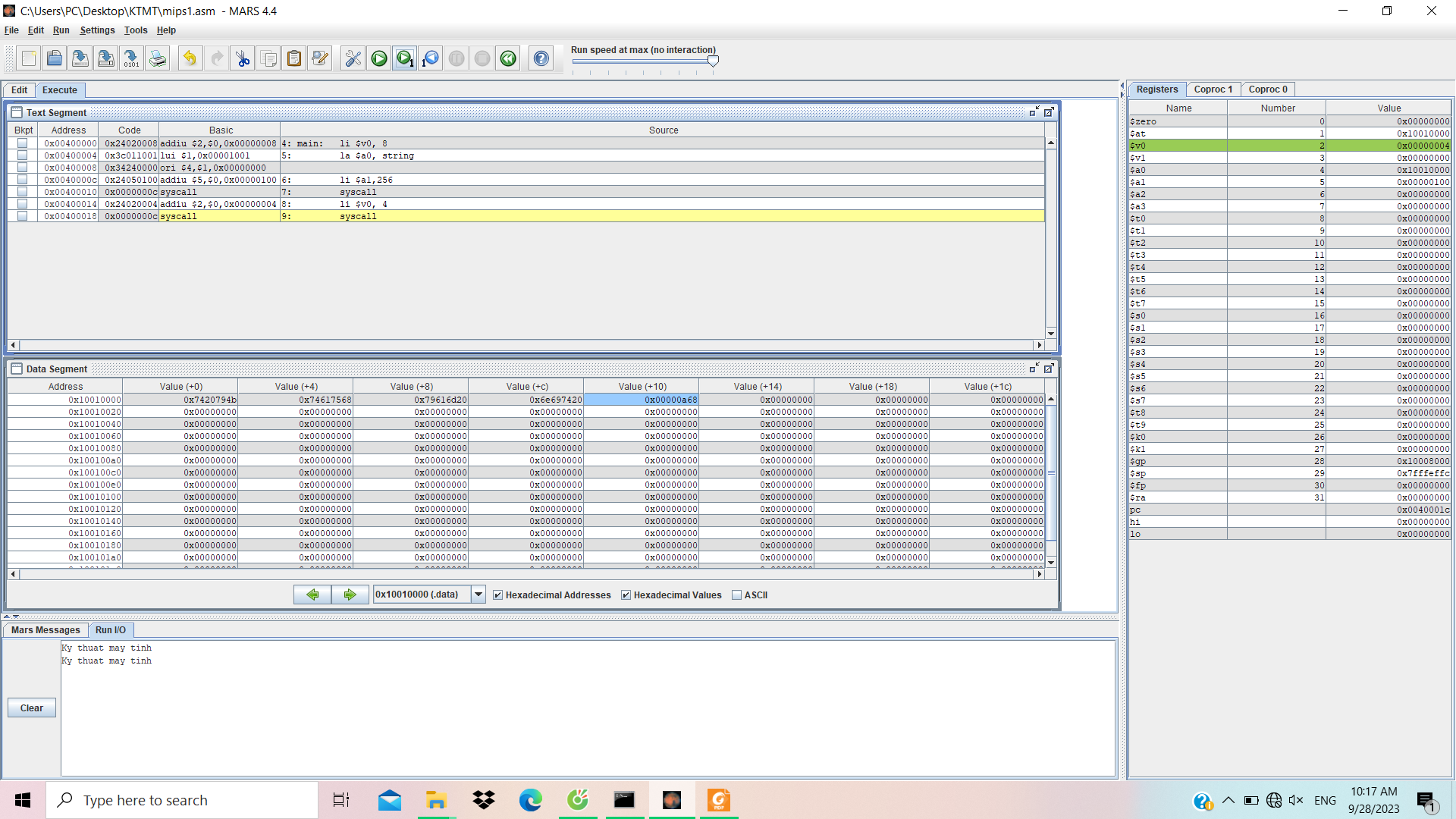
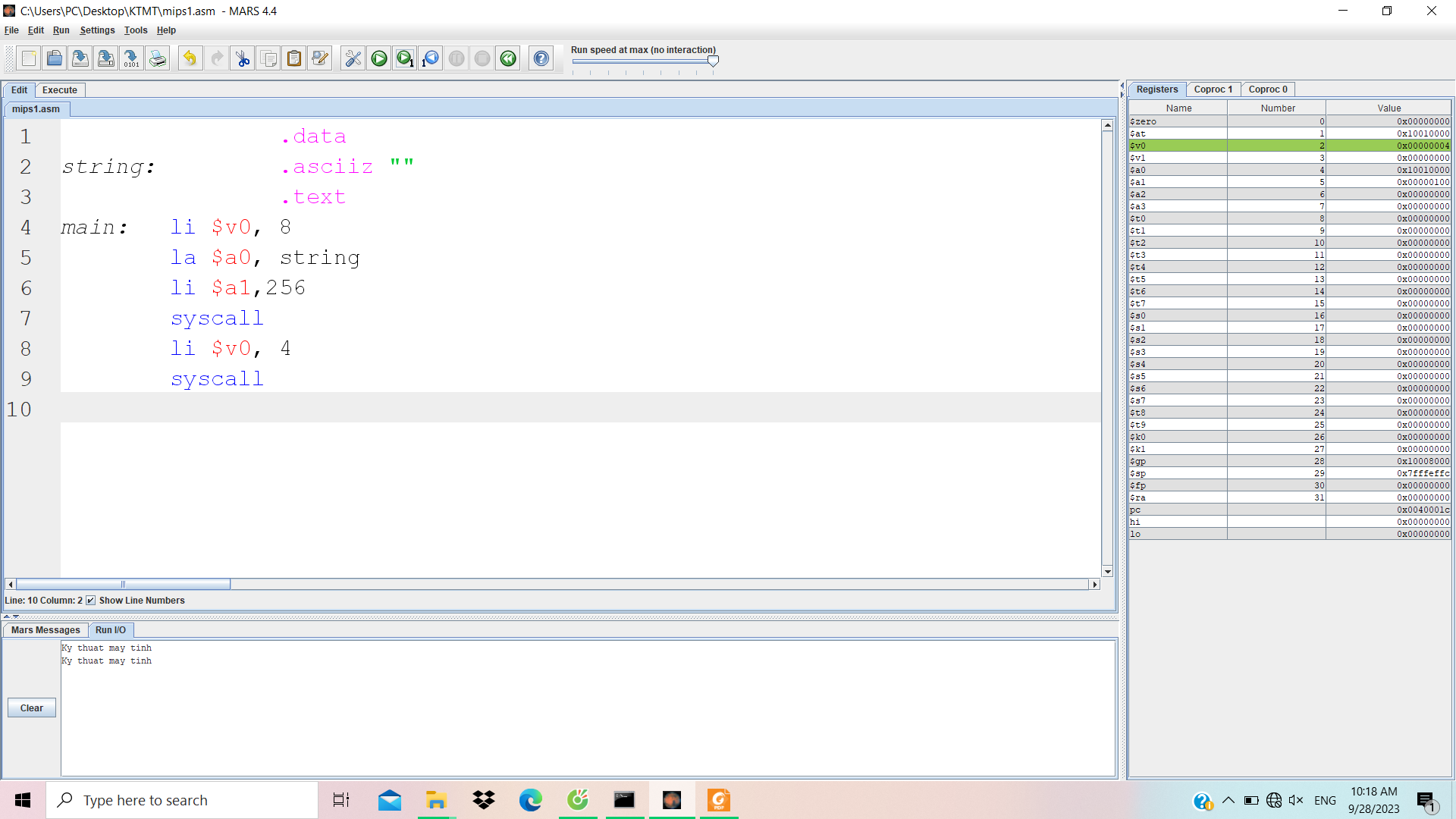


Đầu tiên em khai báo hai biến string1 và string2 với nội dung là hai chuỗi đề bài cho, riêng string1 em để \n đằng sau để nó được cách dòng (vì vậy chuỗi mã hex cuối của string1 sẽ là 0x0a\* thay vì 0x00\*). Để sử dụng chức năng in chuỗi của syscall em phải gán $v0 bằng 4, rồi lần lượt gán địa chỉ của từng chuỗi vào $a0 và gọi syscall

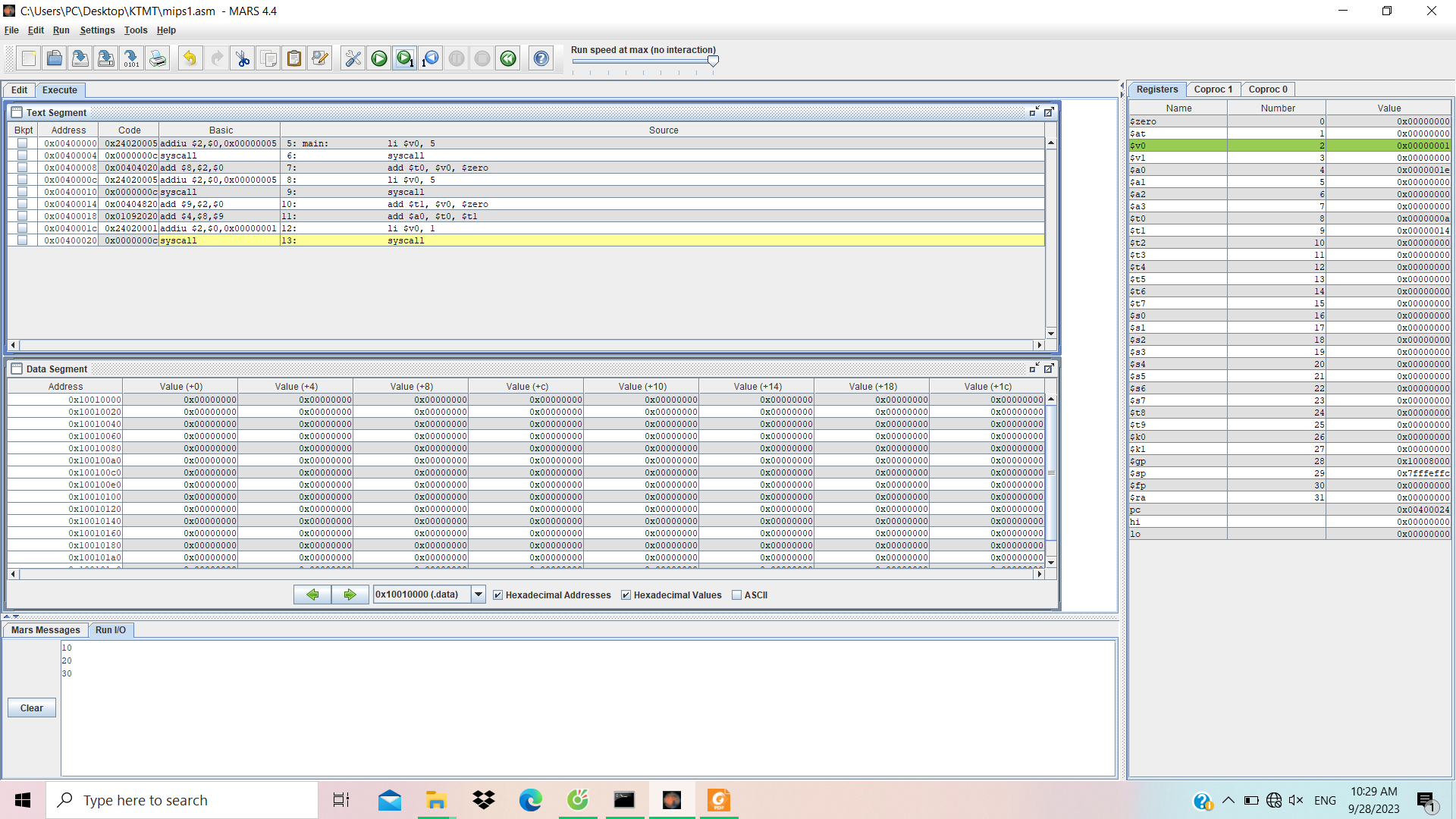
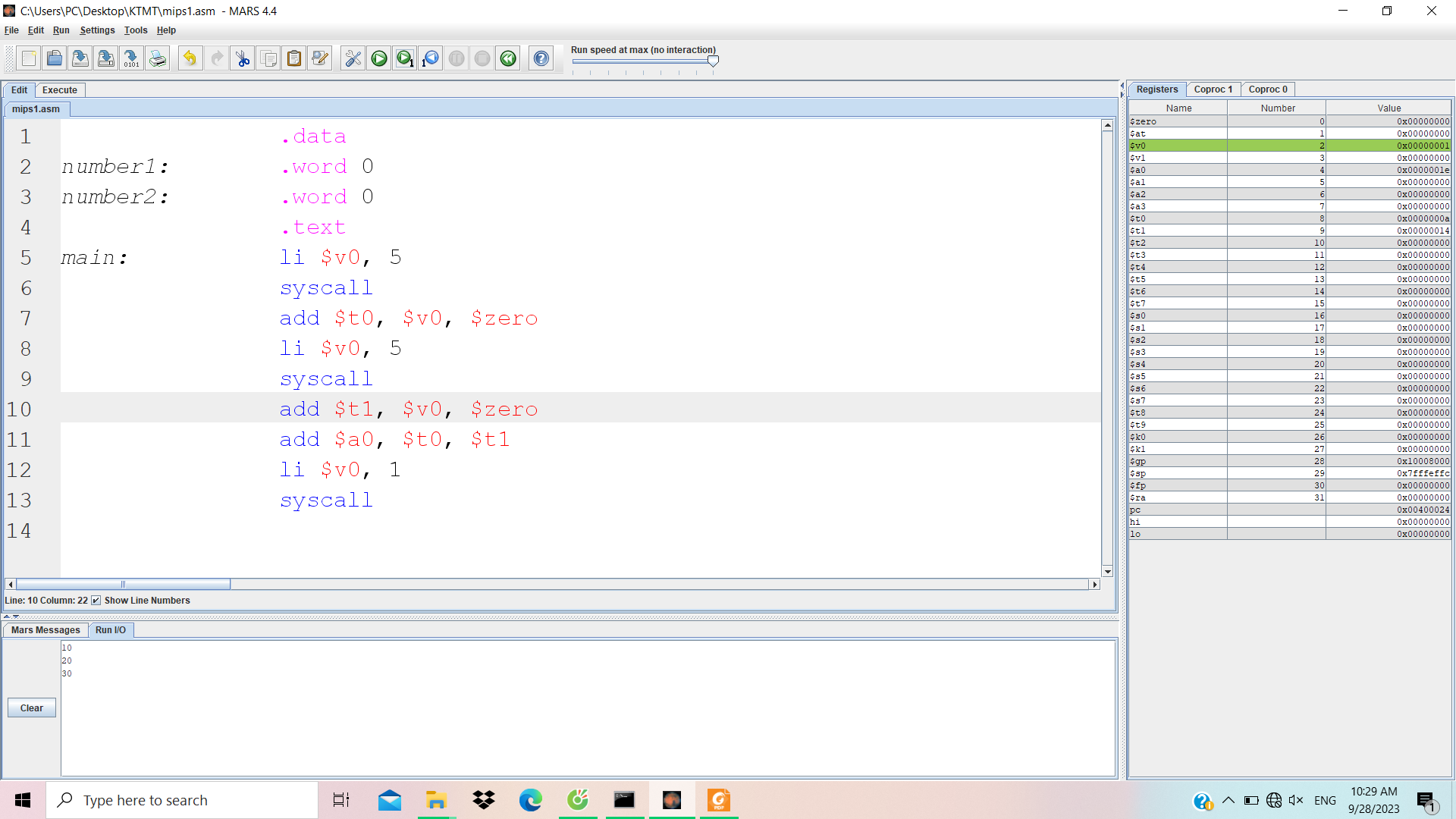
**b)** chuỗi 1: 0x6f616843 0x6e616220 0x61422021 0x616c206e 0x6e697320 0x69762068 0x6e206e65 0x74206d61 0x6d207568 0x003f7961

chuỗi 2: 0x68694800 0x6d202c69 0x20686e69 0x7320616c 0x20686e69 0x6e656976 0x6d616e20 0x75687420 0x5e203120 0x00005e2d

**c)**

****

Bài này thì em khai báo biến string rỗng. Để dùng chức năng lưu chuỗi từ input của syscall em cho $v0 bằng 8, gán địa chỉ string vào $a0 và gán độ dài tối đa của chuỗi vào $a1 (cái này em gán đại 256 để lưu được cho nhiều) rồi gọi syscall. Sau khi đã lưu chuỗi em sử dụng chức năng in chuỗi của syscall bằng cách gán 4 cho $v0 ($a0 thì không cần gán địa chỉ string nữa vì đã làm ở trên rồi) và gọi syscall

**d)  
**

Ở bài này em khai báo hai biến number1 và number2 với giá trị ban đầu là 0, sau đó em sử dụng chức năng lưu số nguyên của syscall bằng cách cho $v0 bằng 5 và gọi syscall. Do chức năng này nó lưu thẳng vào $v0 nên để đưa về biến number1 em sử dụng lệnh add $v0 với $zero và lưu vào biến number1 rồi cho $v0 bằng 5 và gọi syscall. Sau đó lại đưa về biến number2 như biến number1. Để tiến hành xuất kết quả ra I/O của syscall thì em gán $v0 bằng 1, do syscall sẽ xuất giá trị của $a0 nên câu lệnh trước em lưu kết quả add của number1 và number2 vào $a0. Cuối cùng thì em gọi syscall để in kết quả