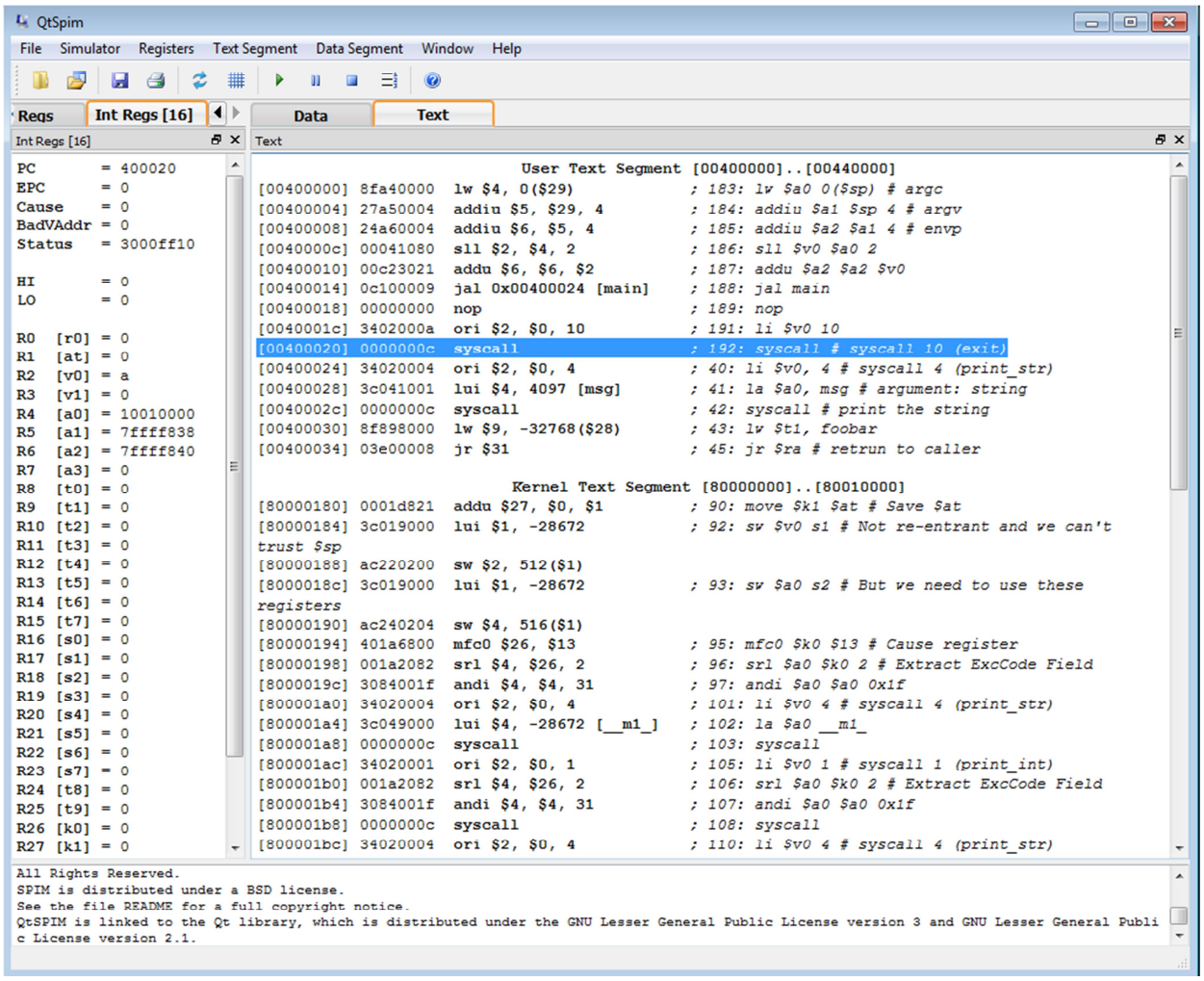
BUỔI THỨ 4

Thời lượng: 3 tiết.

Biên soạn	*Trần Trung Tín*
Phiên bản	21.08.2016
Mọi góp ý đều được đón nhận chân thành, xin gửi email cho tôi đến tttin@tdt.edu.vn	

0. QTSpim

- Đối với máy dùng Win XP thì sử dụng PCSpim.



- Bên trái: tập thanh ghi và giá trị hiện tại.
- Bên phải: Text: mã chương trình.
- Bên phải: Data: dữ liệu chương trình.
- Nạp file mới: File \ Open.
- Chạy toàn bộ code: Simulator \ Run (F5)
- Chạy từng lệnh: Simulator \ Step (F10)
- Chạy lại đoạn code: Refesh (xóa giá trị thanh ghi, mã chương trình vẫn còn lưu trữ).
- Chạy chương trình khác, hay sau khi chỉnh sửa code: cần reinitialize (xóa cả thanh ghi lẫn code)

1. Hello world

- Xuất một chuỗi kí tự ra màn hình, chứ ý mã 4 gán cho $v0 và msg truyền vào $a0.

```
        .data
msg:   .asciiz "Hello TDT"
      .extern foobar 4

        .text
        .globl main
main:   li $v0, 4       # syscall 4 (print_str)
        la $a0, msg     # argument: string
        syscall         # print the string
        lw $t1, foobar

        jr $ra          # retrun to caller
```

Bài tập:
1a, thay đổi thành chuỗi khác và chạy lại đoạn code.
1b, xuất 2 chuỗi "Hello" và "TDT" trên 2 dòng màn hình.

2. Input / Output

- Xuất một giá trị nguyên, chứ ý mã 1 gán cho $v0 và thanh ghi truyền vào $a0.
- Nhập một giá trị nguyên từ bàn phím, mã 5 được dung cho $v0.

```
      # Start .text segment (program code)
      .text

      .globl     main
main:
      # Print string msg1
      li    $v0,4      # print_string syscall code = 4
      la    $a0, msg1  # load the address of msg
      syscall

      # Get input A from user and save
      li    $v0,5      # read_int syscall code = 5
      syscall
      move  $t0,$v0          # syscall results returned in $v0

      # Print string msg2
      li    $v0,4      # print_string syscall code = 4
      la    $a0, msg2  # load the address of msg2
      syscall

      # Get input B from user and save
      li    $v0,5      # read_int syscall code = 5
      syscall
```

```
        move  $t1,$v0          # syscall results returned in $v0

        # Math!
        add   $t0, $t0, $t1    # A = A + B

        # Print string msg3
        li    $v0, 4
        la    $a0, msg3
        syscall

        # Print sum
        li    $v0,1       # print_int syscall code = 1
        move  $a0, $t0    # int to print must be loaded into $a0
        syscall

        # Print \n
        li    $v0,4       # print_string syscall code = 4
        la    $a0, newline
        syscall

        li    $v0,10          # exit
        syscall

        # Start .data segment (data!)
        .data
msg1:   .asciiz    "Enter A:   "
msg2:   .asciiz    "Enter B:   "
msg3:   .asciiz    "A + B = "
newline:    .asciiz    "\n"
```

Bài tập:
2a, Viết đoạn code cho nhập 3 số nguyên và xuất ra tổng của chúng.
2b, Viết đoạn code nhập 2 số nguyên và xuất ra giá trị tổng, giá trị hiệu của chúng.
2c, Hai đoạn code bên trên có cách thoát (exit) khác nhau, hay so sánh và áp dụng cả 2 cách cho các bài tập về sau.

3. Xem hướng dẫn MIPS Floating Point và thực hiện các yêu cầu
3a, Viết đoạn code cho nhập vào 2 số thực float, tính giá trị tổng, hiệu, tích của chúng.
3b, Với số thực double, kích thước của chúng là 64 bits trong khi một thanh ghi có kích thước là 32 bits, MIPS giải quyết ra sao? Thực hiện lại yêu cầu 3a với số thực kiểu double.

4. Viết đoạn code cho phép nhập vào 2 số nguyên, xuất ra giá trị tích và thương của chúng.
Gợi ý: vai trò thanh ghi hi và lo là gì? Tại sao lệnh add và sub có 3 tham số thanh ghi mà lệnh nhân hay chia chỉ có 2 tham số? Khi chia thì kết quả thương lưu vào đâu và kết quả dư lưu vào đâu?

5*. Khi thực hiện nhân 2 số nguyên, giá trị tích có thể lên đến 64 bits, và lưu trữ trong thanh ghi hi và lo, hãy xuất ra giá trị tích này.