Bài tập

Bài 1. Cho biết nếu sử dụng CPU có ISA kiểu Memory-Memory để tính biểu thức sau thì cần bao nhiêu lần truy cập bộ nhớ? Nếu sử dụng CPU có ISA kiểu General Purpose Register thì cần bao nhiêu lần truy cập bộ nhớ?

A = B + C + B*C;

Bài 2. Trong tập lệnh MIPS không có lệnh nhảy nếu so sánh bằng hoặc khác Zero (mạc dù lệnh này được dùng rất phổ biến). Làm thế nào để hiện thực lệnh này hiệu quả

Bài 3. Tìm số nhỏ nhất trong mảng số nguyên, xuất ra giá trị và index vị trí đầu tiên của số nhỏ nhất.

```
.data
iArraySize: .word   10
iArray:     .word   12, 32, 13, 43, 17, 1, -2, -45, 0, 11
numResult:  .asciiz "Minimum number: "
idxResult:  .asciiz "\nIndex: "
.text
main:
### StartCodeHere:

### EndCodeHere:
   # print your results
   la    $a0, numResult
   li    $v0, 4
   syscall
   move  $a0, $s0
   li    $v0, 1
   syscall
   la    $a0, idxResult
   li    $v0, 4
   syscall
   move  $a0, $s1
   li    $v0, 1
   syscall

   #stop program
   li    $v0, 10
   syscall
#your function start from here
```

Bài 4. Sắp xếp chuỗi số nguyên theo giá trị tăng dần.

```
.data
cSpace:     .asciiz " "
cEndLine:   .asciiz "\n"
iArraySize: .word   10
```

```
iArray:     .word   12, 32, 13, 43, 17, 1, -2, -45, 0, 11

.text
main:
   # print integer array
   lw    $t0, iArraySize    # load size of iArray
   la    $t1, iArray        # Load base address of iArray
   jal   print
### StartCodeHere:

### EndCodeHere:

   # print integer array
   lw    $t0, iArraySize   # load size of iArray
   la    $t1, iArray       # Load base address of iArray
   jal   print

   #stop program
   li    $v0, 10
   syscall

print:                      # print fuction

   add   $t2, $0, $0       # index of iArray
loopPrint:
   beqz  $t0, exitPrint    # Check condition
   li    $v0, 1            # service 1 is print integer
   add   $t3, $t1, $t2     # load desired value into $a0
   lw    $a0, ($t3)
   syscall

   li    $v0, 4
   la    $a0, cSpace       # print space just like separator
   syscall

   addi  $t0, $t0, -1      # decrease loop count
   addi  $t2, $t2, 4       # increase index
   b     loopPrint
exitPrint:
   li    $v0, 4
   la    $a0, cEndLine     # print end line
   syscall
   jr    $ra               # end of print
#your function start from here
```

Bài 5. Dùng MIPS assembly viết hàm tính Fibonacci bằng phương pháp đệ quy, $a0 là N, $v0 là kết quả

Bài 6. Dùng MIPS assembly viết hàm tính Fibonacci bằng phương pháp dùng mảng (quy hoạch động), phần tử X[i] của mảng lưu giữ giá trị F(i). $a0 là N, $v0 là kết quả

Bài 7. Hiện thực hàm cộng hai số và đoạn chương trình gọi hàm này bằng cơ chế truyền thông số qua stack (thông số và kết quả trả về đều qua stack)

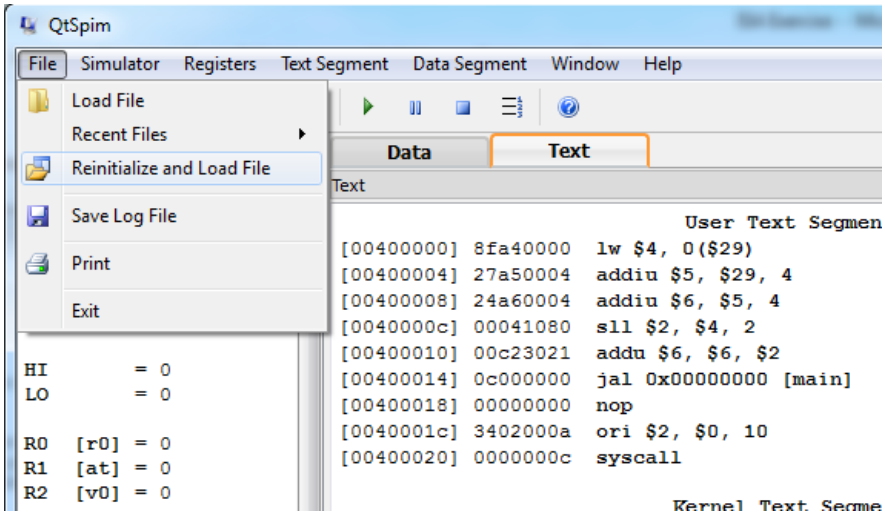
HƯỚNG DẪN MÔ PHỎNG MIPS DÙNG QTSPIM

QtSpim là một chương trình mô phỏng tập lệnh MIPS, đây là phiên bản thay thế cho PCSpim. Phần mềm này sẽ đọc và mô phỏng các file mã nguồn Assembly của MIPS. Các file mã nguồn này được viết dưới dạng file *.S và có format như sau:

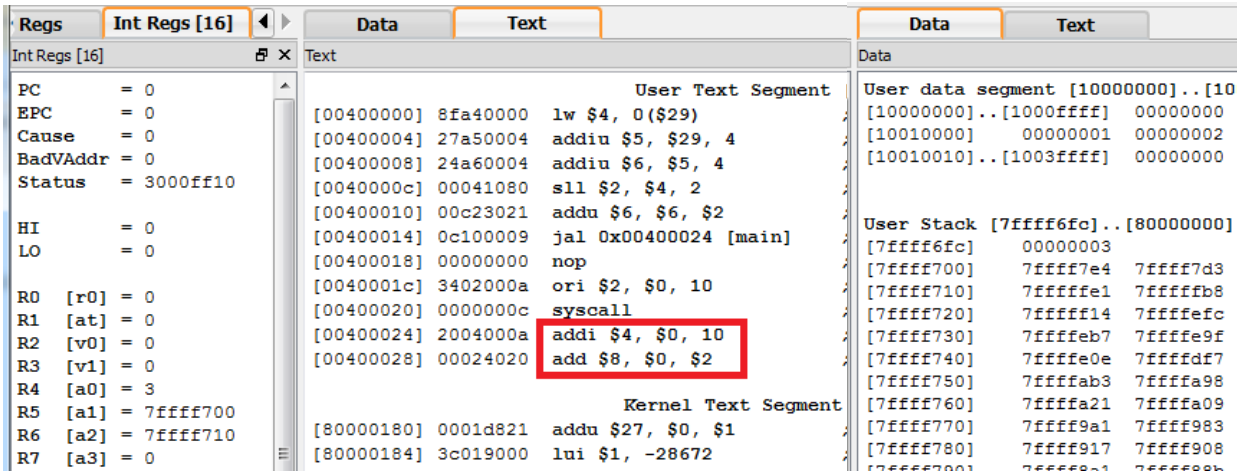
```
# data segment
	.data
	.word 1
	.word 2

# code segment
	.text
	.globl main
main:
	addi $a0, $0, 10
	add $t0, $0, $v0
```

`.data` là khai báo đoạn dữ liệu, là nơi ta khai báo và khởi tạo các biến. `.tex` là khai báo đoạn code, là nơi viết các đoạn mã thực thi. Sau khi cài đặt QtSpim và chạy chương trình QtSpim, vào menu File/Reinitialize and Load File.



Chọn file assembly cần mô phỏng, đoạn chương trình mô phỏng sẽ được load vào hai cửa sổ gồm: Text, hiển thị bộ nhớ chứa lệnh; Data, hiển thị bộ nhớ chứa dữ liệu. Ngoài ra còn một cửa sổ bên phải hiển thị nội dung các thanh ghi.



Khi mô phỏng, giá trị các thanh ghi và các ô nhớ sẽ thay đổi để cho thấy sự thực thi lệnh. Khi load file mã nguồn assembly, QtSpim còn tự động nạp vào các đoạn Kernel code và đoạn code để nhảy đến chương trình người dùng. Đoạn chương trình người dùng sẽ được đặt tại địa chỉ 0x00400024 (khoanh màu đỏ). Sau khi đã nạp file mã nguồn, có thể sử dụng các nút Run, Pause, Stop, Single Step trên thanh công cụ để mô phỏng