Atividades - Assembly RISC-V

- Autor: Rafael Grossi

Atividade 3

Modifique o código do programa anterior para utilizar a instrução mul ao invés da função Multiplica. Você deve remover a função Multiplica do seu código e implementar a multiplicação diretamente na função MultiplicaVetor.

```
.data
vetor: .word 5, 10, 20, 40, 80  # Você pode declarar todos os elemento

.text
main:
    li   a0, 5
    la   a1, vetor   # Pseudo-instrução para carregar o endereço do ve
                     # (no lugar das duas instruções anteriores)
    li   a2, 10
    call MultiplicaVetor

    la   s0, vetor
    li   s1, 5
imprime:
    beq  s1, zero, fimMain
    lw   a1, 0(s0)   # Sintaxe diferente para o lw que era lw a1, s0,
    addi s0, s0, 4
    addi s1, s1, -1
    li   a0, 1
    ecall     # Imprime o número
    li   a0, 11
    li   a1, 13
    ecall     # Imprime uma quebra de linha
    j    imprime

fimMain:
    addi a0, zero, 10
```

```
    ecall   # Encerra a execução do programa

MultiplicaVetor:
    # Movimenta o apontador da pilha 4 posicoes para baixo (16 bytes)
    addi sp, sp, -16
    sw   s0, 12(sp)   # Sintaxe diferente para o sw que era sw s0, sp,
    sw   s1, 8(sp)
    sw   s2, 4(sp)
    sw   ra, 0(sp)

    mv   s0, a0
    mv   s1, a1
    mv   s2, a2

for:
    beq  s0, zero, fim
    lw   a0, 0(s1)
    mv   a1, s2
    # linha alterada
    mul  a0, a0, a1
    sw   a0, 0(s1)
    addi s1, s1, 4
    addi s0, s0, -1
    j    for

fim:
    # Movimenta o apontador da pilha 4 posicoes para cima (16 bytes) e
    lw   ra, 0(sp)
    lw   s2, 4(sp)
    lw   s1, 8(sp)
    lw   s0, 12(sp)
    addi sp, sp, 16
    ret
```

Atividade 4

Agora que você tem mais facilidade para digitar strings, coloque mensagens no seu código para indicar o produto dos números e também para falar que o programa acabou! A ecall de print_string é a 4.

```
.data
    vetor: .word 5, 10, 20, 40, 80  # Você pode declarar todos os elem
    msg1: .string "Multiplicacao = "
    msg2: .string "O Programa acabou!"

.text
main:
    li   a0, 5
    la   a1, vetor   # Pseudo-instrução para carregar o endereço do ve
                     # (no lugar das duas instruções anteriores)
    li   a2, 10
    call MultiplicaVetor

    la   s0, vetor
    li   s1, 5
imprime:
    beq  s1, zero, fimMain
    la a1, msg1
    li a0, 4
    ecall

    lw   a1, 0(s0)   # Sintaxe diferente para o lw que era lw a1, s0,
    addi s0, s0, 4
    addi s1, s1, -1
    li   a0, 1
    ecall     # Imprime o número
    li   a0, 11
    li   a1, 13
    ecall     # Imprime uma quebra de linha
    j    imprime

fimMain:
    la a1, msg2
    li a0, 4
    ecall

    addi a0, zero, 10
    ecall   # Encerra a execução do programa

MultiplicaVetor:
    # Movimenta o apontador da pilha 4 posicoes para baixo (16 bytes)
```

```
    addi sp, sp, -16
    sw   s0, 12(sp)   # Sintaxe diferente para o sw que era sw s0, sp,
    sw   s1, 8(sp)
    sw   s2, 4(sp)
    sw   ra, 0(sp)

    mv   s0, a0
    mv   s1, a1
    mv   s2, a2

for:
    beq  s0, zero, fim
    lw   a0, 0(s1)
    mv   a1, s2
    mul  a0, a0, a1
    mv a1, a0
    sw   a0, 0(s1)


    addi s1, s1, 4
    addi s0, s0, -1
    j    for

fim:
    # Movimenta o apontador da pilha 4 posicoes para cima (16 bytes) e
    lw   ra, 0(sp)
    lw   s2, 4(sp)
    lw   s1, 8(sp)
    lw   s0, 12(sp)
    addi sp, sp, 16
    ret
```

Atividade 5

Experimente desenhar com múltiplas cores no painel. Faça um programa que pinte a tela de branco e depois desenhe um quadrado deixando dois uma borda de dois pontos brancos em todos os lados. Faça esse quadrado trocar de cor múltiplas vezes (você pode adicionar valores nas cores ou armazenar um conjunto de cores num vetor e troca-las). Veja quanto tempo seu simulador gasta para pintar um quadrado e procure ajustar a velocidade do seu programa.

```
.data
# (x << 16) | y
t_left:
    .word 0x00020002
t_right:
    .word 0x00080002
b_left:
    .word 0x00020008
b_right:
    .word 0x00080008
add_col:
    .word 0x00010000
add_row:
    .word 0x00000001

.text

main:
    # clear screen
    li a0, 0x101
    li a1, 0x00FFFFFF
    ecall

    call paint

    li a0, 10
    ecall


paint:
    lw t1, t_left
    lw t2, t_right
    lw t3, b_left
    lw t5, add_col
    lw t6, add_row
    li a0, 0x100

    # row counter
    li s3, 0

    for_v:
```

```
        beq t1, t3, end_v

        # column counter
        li s4, 0
        # current pos(horizontal)
        mv a3, t1

        for_h:
            beq a3, t2, end_h

            # Calculate base color for this row
            li t0, 6
            # s5 = row % 6 (6 colors)
            rem s5, s3, t0

            # RGB components
            li s6, 0
            li s7, 0
            li s8, 0

            beq s5, x0, color_red
            li t0, 1
            beq s5, t0, color_yellow
            li t0, 2
            beq s5, t0, color_green
            li t0, 3
            beq s5, t0, color_cyan
            li t0, 4
            beq s5, t0, color_blue
            j color_magenta

        end_h:
            add t1, t1, t6
            add t2, t2, t6
            addi s3, s3, 1
            j for_v

    end_v:
        ret

color_red:
```

```
    li s6, 255
    li s7, 0
    li s8, 0
    j apply_brightness

color_yellow:
    li s6, 255
    li s7, 255
    li s8, 0
    j apply_brightness

color_green:
    li s6, 0
    li s7, 255
    li s8, 0
    j apply_brightness

color_cyan:
    li s6, 0
    li s7, 255
    li s8, 255
    j apply_brightness

color_blue:
    li s6, 0
    li s7, 0
    li s8, 255
    j apply_brightness

color_magenta:
    li s6, 255
    li s7, 0
    li s8, 255

apply_brightness:
    # brightness = 100 + (s4 * 30); (100 -> 250)
    li t0, 30
    mul t0, s4, t0
    addi t0, t0, 100

    # R
```

```
    mul s6, s6, t0
    li t4, 255
    div s6, s6, t4

    # G
    mul s7, s7, t0
    div s7, s7, t4

    # B
    mul s8, s8, t0
    div s8, s8, t4

    # Combine into RGB
    slli a2, s6, 16
    slli t0, s7, 8
    or a2, a2, t0
    or a2, a2, s8

    # Draw
    mv a1, a3
    ecall

    # increment counter
    add a3, a3, t5
    addi s4, s4, 1
    j for_h
```

Atividade 6

Faça um programa que pinte a bandeira do seu time preferido de futebol no display. Ou outro símbolo do seu agrado!

```
.data
t_left:
    .word 0x00020002
t_right:
    .word 0x00080002
b_left:
    .word 0x00020008
b_right:
    .word 0x00080008
```

```
add_col:
    .word 0x00010000
add_row:
    .word 0x00000001
eye:
    .word 0x000FF000
mouth:
    .word 0x00edbbe2


.text
main:
    li a0, 0x101
    li a1, 0x00FFFFFF
    ecall

    call clearSquare
    call paint

    li a0, 10
    ecall


paint:
    li a0, 0x100
    lw a2, eye

    li a1, 0x00030003
    ecall
    li a1, 0x00030004
    ecall
    li a1, 0x00060003
    ecall
    li a1, 0x00060004
    ecall
    lw a2, mouth
    li a1, 0x00020006
    ecall
    li a1, 0x00030007
    ecall
    li a1, 0x00040007
```

```
    ecall
    li a1, 0x00050007
    ecall
    li a1, 0x00060007
    ecall
    li a1, 0x00070006
    ecall

    ret

clearSquare:
    lw t1, t_left
    lw t2, t_right
    lw t3, b_left
    lw t5, add_col
    lw t6, add_row
    li a0, 0x100
    li a2, 0x000Ab0c0
    for_v:
        beq t1, t3, end_v
        mv a3, t1

        for_h:
            beq a3, t2, end_h

            mv a1, a3
            ecall

            add a3, a3, t5
            j for_h

        end_h:
            add t1, t1, t6
            add t2, t2, t6
            j for_v

    end_v:
        ret
```