175480_04

Patrick de Carvalho Tavares Rezende Ferreira

Atividade 2

1. Ele utiliza o registrador r3 para implementar (atribuir valor) as variáveis e a pilha (stack) para armazená-las.
2. Em r3.
3. Os valores passados para a função são alocados, inicialmente, nos primeiros 4 registradores do hardware e, em seguida, alocados para a pilha (stack). Caso haja mais que 4 parâmetros a serem passados, os restantes são alocados diretamente na pilha.

```
/* This assembly file uses GNU syntax  */

 .text   @ area de código
 .global main  @ torna main visivel para o ambiente de execução

@Funcao inicGPIO não recebe parametros
inicGPIO:
 push  {r7,lr}
 add  r7,sp,#0

@---------------------------------------
@ Escreva aqui o codigo da funcao inicGPIO


 ldr r2, SIM_SCGC5 @ o sinal "=" indica que estamos obtendo o endereço da variavel
SIM_SCGC5, mas queremos seu conteúdo. emtao nao o usaremos.
 ldr r3, =0x00000980 @ salvamos o valor que desejamos passar em r3, pois a instrução str
a seguir nao nos permite passar números imediatos pra algo.
 str r3, [r2,#0] @ Salvamos 980 hexa no próprio SIM_SCGC5.

 ldr r2, PORTC_PCR0 @ O primeiro endereço onde queremos colcoar.
 ldr r3, =0x00000100 @ O valor que quermos passar.

 str r3, [r2, #0] @ Passa 0x00000100 para PORTC_PCR0
 add r2, r2, #4

 str r3, [r2, #0] @ Passa 0x00000100 para PORTC_PCR1
 add r2, r2, #4

 str r3, [r2, #0] @ Passa 0x00000100 para PORTC_PCR2
 add r2, r2, #4

 str r3, [r2, #0] @ Passa 0x00000100 para PORTC_PCR3
 add r2, r2, #4

 str r3, [r2, #0] @ Passa 0x00000100 para PORTC_PCR4
 add r2, r2, #4

 str r3, [r2, #0] @ Passa 0x00000100 para PORTC_PCR5
 add r2, r2, #4

 str r3, [r2, #0] @ Passa 0x00000100 para PORTC_PCR6
 add r2, r2, #4

 str r3, [r2, #0] @ Passa 0x00000100 para PORTC_PCR7
 add r2, r2, #12
```

```
 str r3, [r2, #0] @ Passa 0x00000100 para PORTC_PCR10

 ldr r2, GPIOC_PDOR @ Guarda o conteúdo de GPIOC_PDOR em r2.
 ldr r3, =0 @ O valor que queremos passar.
 str r3, [r2] @ Passa o valor de r3 para o conteúdo da VARIÁVEL GPIOC_PDOR (que é o endereço físico do mpróprio GPIOC_PDOR).

 ldr r2, GPIOC_PDDR @ Guarda o conteúdo de GPIOC_PDDR em r2.
 ldr r3, =0x000007FF @ O valor que queremos passar.
 str r3, [r2] @ Passa o valor de r3 para o conteúdo da VARIÁVEL GPIOC_PDDR (que é o endereço físico do mpróprio GPIOC_PDDR).

 ldr r1, =GPIOC_PCOR @ Guarda o endereço de GPIOC_PCOR
 ldr r2, GPIOC_PCOR @ Guarda o conteúdo de GPIOC_PCOR em r2.
 lsl r3, r2, #10 @ Faz o shift no valor original de GPIOC_PCOR.
 str r3, [r2] @ Atualiza o valor de GPIOC_PCOR.

@--------------------------------------

  cpy  sp,r7 @ ESSA INSTRUÇÃO NAO ESTÁ INVERTIDA?? Funciona pelo fato de r7 nao ter sido utilizado.
 pop  {r7,pc}

@ Funcao delay recebe parametro de espera em r0
delay:
 push  {r7,lr}
 sub  sp,sp,#8
  add  r7,sp,#0
 str  r0,[r7,#4]
   b  testa

volta:
 ldr  r3,[r7,#4]
    sub  r3,#1
    str  r3,[r7,#4]

testa:
 ldr  r3,[r7,#4]
   cmp  r3,#0
  bne  volta

 cpy  sp,r7
  add  sp,sp,#8
   pop  {r7,pc}

@Funcao main
```

```
main:

@------------------------------------
@ Escreva aqui codigo da funcao main

 bl inicGPIO

 @ r5 e r6 serão, respectivamente, i e s.

 for:
  ldr r5, =0x0
  b executa

 teste:
  cmp r5, #255
  ble executa
  b restart


 executa:
  ldr r1, =GPIOC_PDOR @ Guarda endereço de GPIOC_PDOR em r1.
  ldr r6, GPIOC_PDOR @ Guarda valor de GPIOC_PDOR em r9.
  ldr r4, =0xFFFFFF00 @ Determina a máscara do AND.
  and r6, r6, r4 @ Realiza o and e salva o resultado em s (r6).
  orr r6, r6, r5 @ Realiza o OR entre s e i, atualizando s.
  ldr r1, [r1] @ Dereferenciamos este ponteiro para obter acesso ao hardware e nao à
variável.
  str r6, [r1] @ Atualiza GPIOC_PDOR.

  ldr r1, =GPIOC_PSOR @ Guarda o endereço de GPIOC_PSOR
  ldr r2, =0x1 @ Guarda o conteúdo de 0x1 em r2.
  lsl r3, r2, #10 @ Faz o shift no valor original de GPIOC_PSOR.
  ldr r1, [r1] @ Dereferenciamos este ponteiro para obter acesso ao hardware e nao à
variável.
  str r3, [r1] @ Atualiza o valor de GPIOC_PSOR.

  ldr r1, =GPIOC_PSOR @ Guarda o endereço de GPIOC_PCOR
  ldr r2, =0x1 @ Guarda o conteúdo de 0x1 em r2.
  lsl r3, r2, #10 @ Faz o shift no valor original de GPIOC_PCOR.
  ldr r1, [r1] @ Dereferenciamos este ponteiro para obter acesso ao hardware e nao à
variável.
  str r3, [r1] @ Atualiza o valor de GPIOC_PCOR.

  add r5, r5, #1

  ldr r0, CONST_DELAY
```

```
  bl delay

  b teste

 restart:
  b for
@------------------------------------

 .align 2
SIM_SCGC5:   .word 0x40048038
PORTC_PCR0:  .word 0x4004B000
PORTC_PCR1:  .word 0x4004B004
PORTC_PCR2:  .word 0x4004B008
PORTC_PCR3:  .word 0x4004B00C
PORTC_PCR4:  .word 0x4004B010
PORTC_PCR5:  .word 0x4004B014
PORTC_PCR6:  .word 0x4004B018
PORTC_PCR7:  .word 0x4004B01C
PORTC_PCR10: .word 0x4004B028
GPIOC_PDOR:  .word 0x400FF080
GPIOC_PSOR:  .word 0x400FF084
GPIOC_PCOR:  .word 0x400FF088
GPIOC_PDDR:  .word 0x400FF094
CONST_7FF:   .word 0x000007FF
CONST_DELAY: .word 500000
```