GRADO EN INGENIERÍA INFORMÁTICA EN SISTEMA DE INFORMACIÓN

Fundamentos de Computadores

COMPONENTES DEL GRUPO DE TRABAJO
- Ricardo Álamo Fernández
- Rodrigo Bautista Hernández
-

EPD 4

Se incluyen a continuación algunos comentarios/criterios que deberá tener presente siempre que realice ejercicios de pila/subrutinas:

NOTA1

Recuerde que a no ser que se diga lo contrario en el ejercicio en cuestión, el uso de subrutinas y pila debe respetar los siguientes criterios cumpliendo el estándar AAPCS:

- Toda la gestión relativa a la subrutina (parámetros de entrada / salida, preservar valores de registros en programa principal) debe realizarse a través de la pila.
- La pila debe quedar en el mismo estado que tenía antes de la llamada a la subrutina desde el programa principal.
- Además de la codificación de la **SUBRUTINA** solicitada, la resolución del ejercicio debe incluir el código del **PROGRAMA PRINCIPAL**, que necesariamente constará de
    - la **LLAMADA** a la subrutina
    - la posterior **GESTIÓN DEL RESULTADO** devuelto por la subrutina al programa principal, que debe ser asignado a la variable del programa principal que corresponda.

NOTA2 (aplica a CAPTURAS_PANTALLA)**

En los ejercicios en que se haga uso de la pila, la zona de memoria donde se ubican los parámetros/registros/variables apilados en la pila del sistema, debe mostrarse claramente en las capturas de pantalla.

Además, esta captura de pantalla debe ir acompañada de un dibujo/esquema que demuestre que se entiende el contenido de la pila: qué parámetros/registros/variables hay y en qué ubicación.

Todo lo anterior, debe realizarse AL MENOS en el momento de la ejecución del programa en el que la pila tenga apilados todos los parámetros/registros/variables utilizados en el ejercicio en cuestión.

C1 SUBRUTINAS Y PILA

Ejercicio de simulación y comprensión de un ejemplo de uso de Subrutinas y Pila (*subrutina para sumar un vector de números de tipo Word)*. Descargue el archivo correspondiente en Aula Virtual, ensámblelo y ejecútelo en la Raspberry. Los parámetros que se usan son:

- R0: Puntero al vector a sumar.
  Como salida: suma del vector.
- R1: Tamaño del vector.

ARCHIVOS RESULTADO**: EPD4_C1.S y CAPTURAS_PANTALLA****

C2 SUBRUTINAS Y PILA. DEPURACIÓN

Realice una subrutina que reciba 3 vectores, cuyos componentes son tipo Byte, y el tamaño de lo vectores como parámetros de entrada y devuelva como resultado la suma de ellos. Descargue el archivo correspondiente en Aula Virtual, ensámblelo y ejecútelo en la Raspberry.

```
.text
.global main
main:       ldr    r4,=resul    /* r4 puntero al resultado */
            ldr    r0,=tam
            ldrb   r0,[r0]
            ldr    r1,=vec1
            ldr    r2,=vec2
            ldr    r3,=vec3
            stmdb  sp!, {r4}      /* Pasamos el vector resultado por pila */
            bl     suma_vec     /* Llamada a la subrutina suma_vec */
            ldmia  sp!,{r4}      /* Quitamos el parámetro anterior */
            mov    lr,#0
            bx     lr
/* Subrutina que suma tres vectores de bytes */
/* Parametros de entrada: r0 (tamaño), r1-r3 (punteros entrada ), por pila: puntero a vector suma*/
/* Parametros de salida: No existen */
/* Variables locales: r4 (puntero al resultado) */
/*                    r5 (sumas parciales) */
/*                    r6 (carga de datos) */
suma_vec:
            stmdb  sp!,{r4,r5,r6}
            ldr r4,[sp,#12]

suma_vec_bucle:
            subs   r0, r0, #1                  /* r1 <-- r1 - 1 */
            blt    fin_bucle
            mov    r5,#0  /* Inicializamos el r2 para sumar */
            ldrb   r6, [r1, r0]   /* r6 <-- *(r0 + r1) */
            add    r5, r5, r6   /* r5 += r6 */
            ldrb   r6, [r2, r0]   /* r6 <-- *(r2 + r0) */
            add    r5, r5, r6   /* r5 += r6 */
            ldrb   r6, [r3, r0]   /* r6 <-- *(r3 + r0) */
            add    r5, r5, r6   /* r5 += r6 */
            strb   r5, [r4, r0]             /* *(r4+r0) <-- r5 */
            b      suma_vec_bucle

fin_bucle:
            ldmia  sp!, {r4,r5, r6}    /* Recuperamos los valores guardados*/
            bx     lr
```

ARCHIVOS RESULTADO**: EPD4_C2.S y CAPTURAS_PANTALLA****

C3 SUBRUTINAS Y PILA. DEPURACIÓN

En un el archivo EPD4_C3_bug.s se propone una subrutina. Se pide:

1. Indique cuál es su función.

- **Buscar el número mas grande**

2. Indique los parámetros de entrada y de salida de la misma, y cómo se pasan a la función.

- **parametro entrada: vector “nums” // parametro salida: “resul”**

3. Corrija los fallos existentes en dicha función.

```
.data
nums:       .word   32,45,78,91,12
resul:      .word   -1

.text
.global main
main:
    ldr r5,=nums
    ldmia r5,{r0-r4}
    stmdb sp!, {r0-r4}      /*camio stmda a stmdb y cambiamos los registros a r0-r4*/
    ldr r6,=resul           /*línea nueva*/
    bl subrut
    ldm sp, {r0}            /*línea nueva*/
    str r0, [r6]            /*línea nueva*/
    mov lr,#0
    bx lr


subrut:
    ldmia sp!,{r0-r4}       /*línea nueva*/
    cmp r0,r1
    movlt r0, r1
    cmp r0,r2
    movlt r0, r2
    cmp r0,r3
    movlt r0, r3        /*cambiamos r2 por r3*/
    cmp r0,r4
    movlt r0,r4
    stmdb sp!,{r0}      /*línea nueva*/
    bx lr
```

Realice dibujos de la pila en la versión correcta para verificar que se han corregido los fallos. Puntos representativos donde hay que mostrar la pila:

a) Antes de llamar a la subrutina (bl …).

a) (Valores del vector "nums", apilados)

Sp→ 00000020
0000002d
0000004e
0000005b
0000000c

b) Al principio de la subrutina (justo después de ejecutar la instrucciones que preservan los registros del programa principal).

b) (Después de desapilar los valores del vector "nums")

00000020
0000002d
0000004e
0000005b
0000000c
Sp→ 00000000

c) Antes de la instrucción de retorno de la subrutina (bx lr …) .

c) (Los valores apilados anteriormente ya no importan así que se sobreescribe el último para guardar el valor que queremos guardar, el número más grande de "nums")

00000020
0000002d
0000004e
0000005b
Sp→ 0000005b

d) En el programa principal, después de recuperar datos.

d)

00000020
0000002d
0000004e
0000005b
Sp→ 0000005b

```
.data
nums:   .word   32,45,78,91,12
resul:  .word   -1

.text
.global main
main:
        ldr r5,=nums
        ldmia r5,{r0-r4}
        stmda sp!,{r4}
        bl subrut
        mov lr,#0
        bx lr

subrut:
        cmp r0,r1
        movlt r0, r1
        cmp r0,r2
        movlt r0, r2
        cmp r0,r3
        movlt r0, r2
        ldr r4, [sp, #4]
        cmp r0,r4
        movlt r0,r4
        bx lr
```

ARCHIVOS RESULTADO**: EPD4_C3.s (corregida) y CAPTURAS_PANTALLA****

C4 SUBRUTINAS Y PILA

Se tiene un vector de números **VEC** de tamaño **TAM**. Se pide realizar una **SUBRUTINA** que obtenga el elemento de valor mínimo en dicho vector.

Los parámetros de entrada de dicha subrutina deben ser la dirección del primer elemento del vector y el tamaño del vector. El resultado debe quedar almacenado por el programa principal en la variable **MIN**.

ARCHIVOS RESULTADO: **EPD4_C4.s y CAPTURAS_PANTALLA****

```
.data
vec:    .word   5,10,7,2,8
TAM:    .word   5
min:    .word   -1

.text
.global main
main:
        ldr r0,=vec
        ldr r1,=TAM
        ldr r1,[r1]
        stmdb sp!,{r0-r1}       /* Apila los parámetros de entrada */
        bl subrut
        ldmia sp!,{r3}
        ldr r4,=min
```

```
        str r3, [r4]
        mov lr,#0
        bx lr


subrut: ldmia sp!,{r0-r1}/* Desapila los parámetros de entrada */
        ldr r3,[r0]             /* Inicializa el valor de r3 */
buc:    cmp r1,#0               /* Contador del bucle */
        beq fbuc
        ldr r2,[r0],#4
        cmp r2,r3
        movlt r3,r2             /* Si el valor de vec cargado en r2 es menor que el que hay en r3, r2->r3 */
        sub r1,r1,#1            /* Decrementa el contador */
        b buc
fbuc:   stmdb sp!,{r3}          /* Apila el parámetro salida */
        bx lr
```

C5 SUBRUTINAS. CADENAS DE CARACTERES (I)

Realice una función que calcule la longitud de una cadena de caracteres. Proponga los registros que se usarán como entrada y salida. Demuestre su funcionamiento obteniendo la longitud de tres cadenas diferentes y almacenando los resultados en memoria. Datos en memoria del programa principal:

cadena1: .asciz "prueba"

cadena2: .asciz "FC no es tan dificil"

cadena3: .asciz "Me encanta el ensamblador"

.align

resul1: .word -1

resul2: .word -1

resul3: .word -1

ARCHIVOS RESULTADO: **EPD4_C5.s y CAPTURAS_PANTALLA****

```
.data

cadena1:    .asciz "prueba"
cadena2:    .asciz "FC no es tan dificil"
cadena3:    .asciz "Me encanta el ensamblador"

        .equ nul,0

.align

resul1: .word -1
resul2: .word -1
resul3: .word -1

.text
.global main

main:   ldr r0,=cadena1
```

```
        ldr r1,=resul1
        stmdb sp!,{r0-r1}       /* Carga la dirección de cadena1 y resul1 en la pila */
        bl subrut
        ldr r0,=cadena2
        ldr r1,=resul2
        stmdb sp!,{r0-r1}       /* Carga la dirección de cadena2 y resul2 en la pila */
        bl subrut
        ldr r0,=cadena3
        ldr r1,=resul3
        stmdb sp!,{r0-r1}       /* Carga la dirección de cadena3 y resul3 en la pila */
        bl subrut
        mov lr,#0
        bx lr


subrut: ldmia sp!,{r0-r1}               /* Desapila las direcciones */
buc:    ldrb  r2, [r0], #1
        add   r3, r3, #1        /* Contador de elementos de la cadena */
        cmp   r2, #nul          /* Comprueba si se ha acabado la cadena */
        bne   buc
        sub   r3, r3, #1        /* Decrementa el contador si el último elemento que se ha cargado es
nulo */
        str   r3,[r1]                   /* Guarda el contador en la dirección en r1 */
        bx lr
```

C6 SUBRUTINAS Y PILA. CADENAS DE CARACTERES (II)

Realice una función que concatene dos cadenas de caracteres teniendo un valor máximo de caracteres. Dicha función tiene como parámetros:

1. R0: (IN-OUT) Tamaño máximo de la cadena concatenada. Salida: longitud de la cadena.
2. R1: (IN) Dirección de cadena de entrada 1
3. R2: (IN) Dirección de cadena de entrada 2
4. R3: (IN) Dirección de cadena de salida

Datos en memoria del programa principal:

```
cadena1: .asciz "Salvaguardar"

cadena2: .asciz " es necesario"

cadena3: .spaces 30

.align

tam:     .word 29

resul:   .word -1
```

ARCHIVOS RESULTADO: **EPD4_C6.s y CAPTURAS_PANTALLA****

```
.data
```

```
cadena1:        .asciz "Salvaguardar"
cadena2:        .asciz " es necesario"
cadena3:        .space 30

        .equ nul,0

.align

tam:    .word 29
resul:  .word -1

.text
.global main

main:   ldr r0,=tam
                ldr r1,=cadena1
                ldr r2,=cadena2
                ldr r3,=cadena3
                stmdb sp!,{r0-r3}       /* Apila los parámetros de entrada */
                bl subrut
                ldmia sp!,{r0}          /* Desapila el parámetro de salida */
                ldr r9,=resul
                str r0,[r9]             /* Guarda el resultado en resul */
                mov lr,#0
                bx lr                   /* Fin */


subrut:  ldmia sp!,{r0-r1}          /* Desapila los parámetros de entrada*/
                mov r4,#0
buc:    cmp r4,r0                       /* Comprueba que no se haya sobrepasado el número máximo de
elementos de cadena3 */
                beq fin
                ldrb r5,[r1],#1
                add r4,r4, #1           /* Contador de elementos de la cadena */
                cmp r5,#nul             /* Comprueba si se ha acabado la cadena */
                beq buc2
                strb r5,[r3],#1         /* Guarda un elemento de cadena1 en cadena3 y actualiza la
dirección de cadena3 */
                b buc
buc2:   cmp r4,r0                       /* Comprueba que no se haya sobrepasado el número máximo de
elementos de cadena3 */
                beq fin
                ldrb r5,[r1],#1
                add r4,r4,#1            /* Contador de elementos de la cadena */
                cmp r5,#nul             /* Comprueba si se ha acabado la cadena */
                beq fin
                strb r5,[r3],#1         /* Guarda un elemento de cadena1 en cadena3 y actualiza la
dirección de cadena3 */
                b buc2

fin:            mov r0,r4               /* Guarda el tamaño final de la cadena3 en r0 */
                stmdb sp!,{r0}          /* Apila los parámetros de salida */
                bx lr
```

Anexo: Procedimiento estándar de llamadas a subrutinas

Dividimos dicho procedimiento en de la invocación de la subrutina (parte llamante) y el procedimiento interno de la función.

Invocación de la función:

1. (opcional) **Paso de parámetros del quinto en adelante por pila**
   (normalmente se usan r0-r3 como vars temporales)
   (sólo para funciones de más de 4 elementos)
2. **Escritura de parámetros r0-r3**
3. **Llamada a subrutina (bl subrut)**
4. (opcional) **Sumar al sp 4*N** (N=nº de parámetros apilados en el paso 1)

Procedimiento de la subrutina:

1. (opcional 1) **Salvaguardar los registros** que se empleen:
   - De r4 a r10
   - lr si hacemos dentro otra llamada a subrutina
2. (opcional 2) Decrementar la pila para variables locales.
3. **Cuerpo de la función**.
4. Se devuelve el valor resultado en r0 (ó en r1:r0 si el resultado es doble palabra, 8 bytes).
5. (opcional 2) Incrementar la pila para liberar variables locales definidas en el paso 2.
6. (opcional 1) Restaurar los registros salvaguardados en el paso 1.
7. **Retornar** la función con bx lr.

NOTA: La parte opcional 2 de la función raramente se usa (si se llega a usar) en esta asignatura.