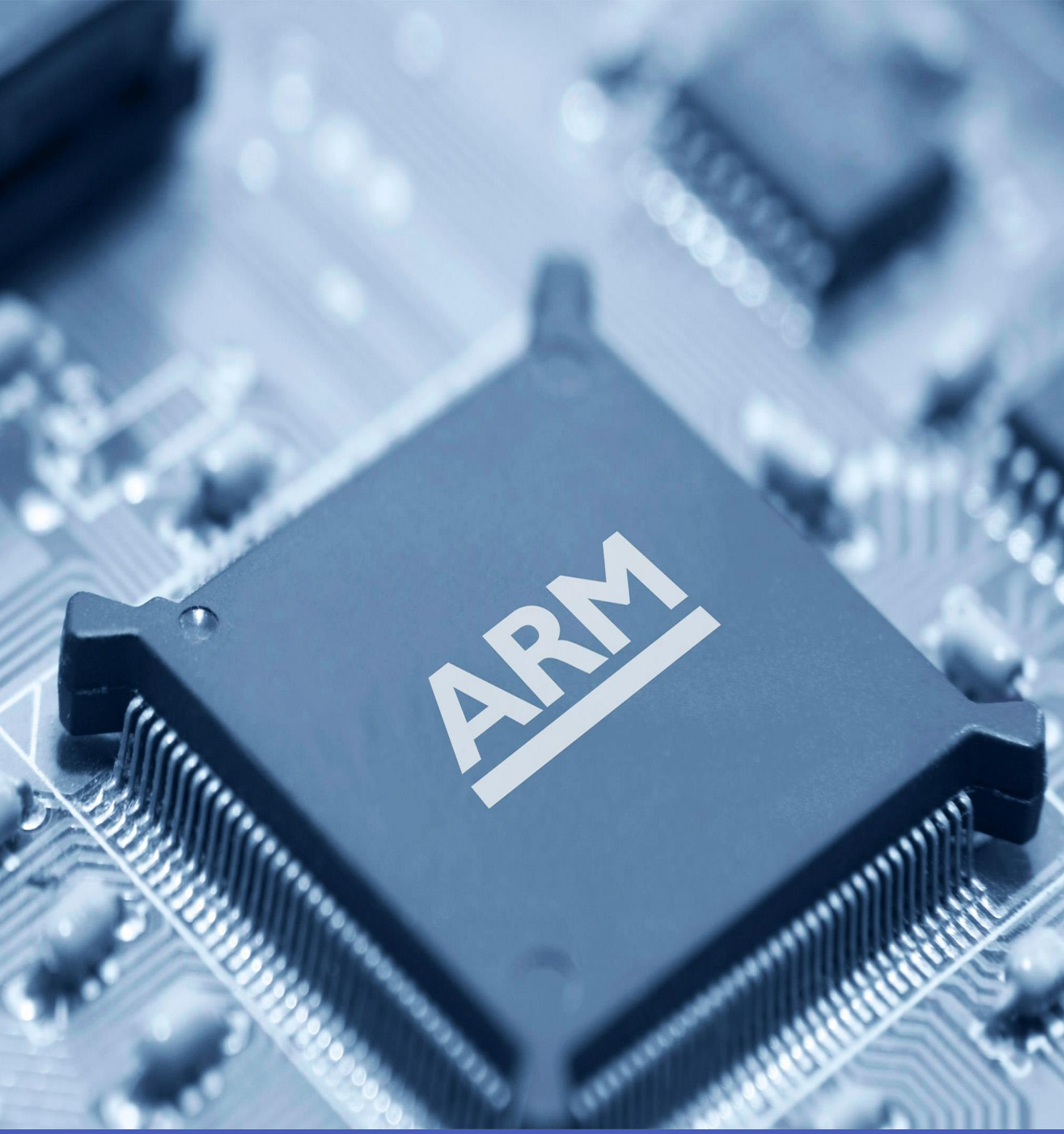

TRABAJO ARM

Jesús de la Higuera Cuesta – Grupo D3

EJERCICIO 1

Desarrollar un programa en el que:

- Se declara en memoria una cadena de 20 caracteres + terminador.
- Se declara en memoria una constante *c* de tamaño byte y valor = 21 decimal.
- Se reserva espacio en memoria para otra cadena del mismo tamaño exacto (cuidado de no descuadrar la alineación del resto de los datos e instrucciones en memoria).
- Mediante un bucle simple, procesar los caracteres de acuerdo con la siguiente operación:
    - destino[i] = origen[i] XOR *c*.
- La cadena destino se imprime por pantalla.

TÉCNICAS UTILIZADAS

Mediante el uso de un bucle do-while se recorre la posición i de la cadena origen en cada iteración, cargando el valor de dicha posición y de la constante c en registros para operar con ellos, guardando finalmente el resultado de la operación en la posición i de la cadena destino antes de pasar a la siguiente iteración con la siguiente posición de la cadena origen y destino.

PROBLEMAS ENCONTRADOS

Hubo un descuadre de la alineación de algunos datos e instrucciones en memoria debido a la falta de la directiva ALIGN, que produjo resultados erróneos al imprimir la cadena resultante

También hubo dificultades para realizar la operación XOR por el desconocimiento de esta operación en ARM.

SOLUCIONES ADOPTADAS

Usar la directiva ALIGN después de la declaración en memoria de la constante y las cadenas solucionó el error del descuadre.

Respecto a la operación XOR, se solucionó en clase.

EJERCICIO 1-B

Mejorar el programa anterior, utilizando las técnicas de la práctica 3, para que el programa sea lo más eficiente posible (mínimo número de instrucciones y de ciclos de ejecución). Añadir comentario explicando qué técnica se ha usado y por qué mejora el programa.

TÉCNICAS UTILIZADAS

El funcionamiento del programa es similar al del ejercicio 1, a diferencia de que en este reduzco la cantidad de código utilizando la técnica de post-indexado en el acceso a la posición i de la cadena de origen y en la escritura del valor final en la posición i de la cadena destino, ahorrándome así escribir las instrucciones **ADD r2,r2,#1** y **ADD r3,r3,#1** , respectivamente.

También extraigo la instrucción **LDRB r4,[r1]** (correspondiente a cargar el valor de c en r4) del bucle ya que es innecesario que se realice en cada iteración, porque el valor de c va a permanecer constante almacenado en el registro r4.

PROBLEMAS ENCONTRADOS

No se encontraron más problemas.

EJERCICIO 2

Desarrollar un programa en el que:

- Se declara en memoria una cadena de 20 caracteres + terminador.
- El programa pasa a una rutina dos parámetros:
  - r11 = puntero al inicio de la cadena.
  - r12 = número de caracteres (excluido el terminador).
- La rutina calcula el máximo (es decir, el carácter con el máximo valor ASCII) entre los elementos de la cadena y deja el resultado en el registro ro.
- Se valorará el uso de técnicas avanzadas para hacer más eficiente el programa.

TÉCNICAS UTILIZADAS

Mediante el uso de un bucle do-while se recorre la posición i de la cadena en cada iteración, comprobando si el carácter de esta es superior al carácter máximo (almacenado en el registro r0). En caso afirmativo, se salta a la etiqueta MAXIMO, encargada de almacenar en r0 el carácter de la posición i antes de pasar a la etiqueta ITERA, que aumenta en 1 el contador y comprueba si volver a BUCLE para realizar otra iteración con la siguiente posición de la cadena.

Uso la técnica de post-indexado en la carga del carácter i de la cadena en cada iteración del bucle, ahorrándome así escribir la instrucción **ADD r11,r11,#1**.

PROBLEMAS ENCONTRADOS

Inicialmente hubo problemas almacenando el valor máximo debido a la mala colocación de la etiqueta MAXIMO.

SOLUCIONES ADOPTADAS

Escribir la etiqueta MAXIMO con el salto incondicional a ITERA al final del código impidió que se realizara esta operación en cada iteración del bucle, solucionando el problema.

EJERCICIO 3

Desarrollar un programa en el que:

- Se declara en memoria un vector de 20 enteros de tamaño 4 bytes cada uno.
  - ...y otro vector igual, que hará de destino.
- El programa rellena el vector destino (excepto los dos últimos elementos) mediante la siguiente fórmula:
  - Destino(n) = 2*origen(n+1) - origen(n) - origen(n+2)
- Se valorará el uso de técnicas avanzadas para hacer más eficiente el programa.

TÉCNICAS UTILIZADAS

Comienzo estableciendo r1, r2 y r3 como punteros hacia las posiciones n, n+1 y n+2 de la cadena ORIGEN, respectivamente (siendo n inicialmente la primera posición).

A continuación, mediante el uso de un bucle do-while, se cargan los valores de las posiciones anteriores de la cadena ORIGEN en registros para operar con ellas y almacenar en la posición n de la cadena DESTINO el resultado de la operación 2*origen(n+1) - origen(n) - origen(n+2). Estas operaciones se realizan en cada iteración desde hasta n=18.

Uso la técnica de post-indexado en la carga de los valores n, n+1 y n+2 en sus correspondientes registros y en el almacenamiento del valor resultante en la posición n de la cadena DESTINO, ahorrando así las instrucciones **ADD r1,r1,#1** , **ADD r2,r2,#1** , **ADD r3,r3,#1** y **ADD r4,r4,#1** , respectivamente.

PROBLEMAS ENCONTRADOS

No se encontraron problemas.

CÓDIGO

EJERCICIO 1

```
        AREA    ej1,CODE,READWRITE

SWI_Salir       EQU     &11
SWI_write0      EQU     &2

c               DCB     21          ; Constante c con valor 21
CADENA          DCB     "aaaaaaaaaabbbbbbbbcc", &a, &d,0
CADENA2 % 21                        ; Reserva espacio para CADENA2

        ALIGN

        ENTRY

        MOV r0,#0                   ; r0 es contador a 0
        ADR r1,c                    ; r1 apunta a c
        ADR r2,CADENA               ; r2 apunta a CADENA
        ADR r3,CADENA2              ; r3 apunta a CADENA2


BUCLE   LDRB r4,[r1]                ; cargamos en r4 valor guardado en c (21)
        LDRB r5,[r2]                ; cargamos en r5 caracter de CADENA


        EOR r6,r5,r4                ; r6 = origen[i] XOR c
        STRB r6,[r3]                ; guardamos valor obtenido de r6 en CADENA2

        ADD r2,r2,#1                ; avanzamos 1 caracter de CADENA
        ADD r3,r3,#1                ; avanzamos 1 caracter de CADENA2

        ADD r0,r0,#1                ; contador ++
        CMP r0,#20                  ; contador < elementos cadena?
        BLT BUCLE                   ; salto a BUCLE


        ADR     r0,CADENA           ; r0 apunta a CADENA
        SWI     SWI_write0          ; escribimos por pantalla contenido de CADENA

        ADR     r0,CADENA2          ; r0 apunta a CADENA2
        SWI     SWI_write0          ; escribimos por pantalla contenido de CADENA2

        SWI SWI_Salir               ; Salir


        END
```

EJERCICIO 1-B

```
          AREA      ej1b,CODE,READWRITE

SWI_Salir           EQU       &11
SWI_write0          EQU       &2

c                   DCB       21          ; Constante c con valor 21
CADENA              DCB       "aaaaaaaaaabbbbbbbbcc", &a, &d,0
CADENA2 % 21                              ; Reserva espacio para CADENA2

          ALIGN

          ENTRY

          MOV r0,#0                       ; r0 es contador a 0
          ADR r1,c                        ; r1 apunta a c
          ADR r2,CADENA                   ; r2 apunta a CADENA
          ADR r3,CADENA2                  ; r3 apunta a CADENA2
          LDRB r4,[r1]                    ; cargamos en r4 valor guardado en c (21)

BUCLE
          LDRB r5,[r2],#1                 ; cargamos en r5 caracter de CADENA

          EOR r6,r5,r4                    ; r6 = origen[i] XOR c
          STRB r6,[r3],#1                 ; guardamos valor obtenido de r6 en CADENA2

          ADD r0,r0,#1                    ; contador ++
          CMP r0,#20                      ; contador < elementos cadena?
          BLT BUCLE                       ; salto a BUCLE

          ADR       r0,CADENA             ; r0 apunta a CADENA
          SWI       SWI_write0            ; escribimos por pantalla contenido de CADENA

          ADR       r0,CADENA2            ; r0 apunta a CADENA2
          SWI       SWI_write0            ; escribimos por pantalla contenido de CADENA2

          SWI SWI_Salir                   ; Salir


          END
```

EJERCICIO 2

```
          AREA      ej3,CODE,READWRITE

SWI_Salir           EQU       &11 ; Codigo de impresion de salida del programa(11)
ORIGEN              DCD       1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
DESTINO             DCD       1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20

          ENTRY

          ADR r1, ORIGEN     ; r1 apunta a ORIGEN(n)
          ADR r2, ORIGEN     ; r2 apunta a ORIGEN(n)
          ADD r2,r2,#4       ; r2 apunta a ORIGEN(n+1)
          ADR r3, ORIGEN     ; r3 apunta a ORIGEN(n)
          ADD r3,r3,#8       ; r3 apunta a ORIGEN(n+2)
          ADR r4, DESTINO    ; r4 apunta a DESTINO

          MOV r10,#0         ; r10 es contador a 0
          MOV r11,#18        ; r11 almacena nº elementos a rellenar en destino (podra hasta 18)

BUCLE
          LDR r5,[r1],#4     ; cargamos en r5 valor n de ORIGEN
          LDR r6,[r2],#4     ; cargamos en r6 valor n+1 de ORIGEN
          ADD r6,r6,r6       ; r6 = 2*(n+1)
          LDR r7,[r3],#4     ; cargamos en r7 valor n+2 de ORIGEN

          ADD r8,r5,r7       ; r8 = n + (n+2)
          SUB r9,r6,r8       ; r9 = 2*(n+1) - (n + (n+2))

          STR r9,[r4],#4     ; guardamos valor de r9 en pos n de DESTINO

          ADD r10,r10,#1     ; contador ++
          CMP r10,r11        ; contador < elementos destino ?
          BLT BUCLE          ; salto a BUCLE

          SWI SWI_Salir      ; Salir

          END
```

EJERCICIO 3

```
          AREA      ej3,CODE,READWRITE

SWI_Salir           EQU       &11 ; Codigo de impresion de salida del programa(11)
ORIGEN              DCD       1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
DESTINO             DCD       1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20

          ENTRY

          ADR r1, ORIGEN     ; r1 apunta a ORIGEN(n)
          ADR r2, ORIGEN     ; r2 apunta a ORIGEN(n)
          ADD r2,r2,#4       ; r2 apunta a ORIGEN(n+1)
          ADR r3, ORIGEN     ; r3 apunta a ORIGEN(n)
          ADD r3,r3,#8       ; r3 apunta a ORIGEN(n+2)
          ADR r4, DESTINO    ; r4 apunta a DESTINO

          MOV r10,#0         ; r10 es contador a 0
          MOV r11,#18        ; r11 almacena nº elementos a rellenar en destino (podra hasta 18)

BUCLE
          LDR r5,[r1],#4     ; cargamos en r5 valor n de ORIGEN
          LDR r6,[r2],#4     ; cargamos en r6 valor n+1 de ORIGEN
          ADD r6,r6,r6       ; r6 = 2*(n+1)
          LDR r7,[r3],#4     ; cargamos en r7 valor n+2 de ORIGEN

          ADD r8,r5,r7       ; r8 = n + (n+2)
          SUB r9,r6,r8       ; r9 = 2*(n+1) - (n + (n+2))

          STR r9,[r4],#4     ; guardamos valor de r9 en pos n de DESTINO

          ADD r10,r10,#1     ; contador ++
          CMP r10,r11        ; contador < elementos destino ?
          BLT BUCLE          ; salto a BUCLE

          SWI SWI_Salir      ; Salir

          END
```