Sistemas Embebidos
Práctica N°1

Tomás Vidal

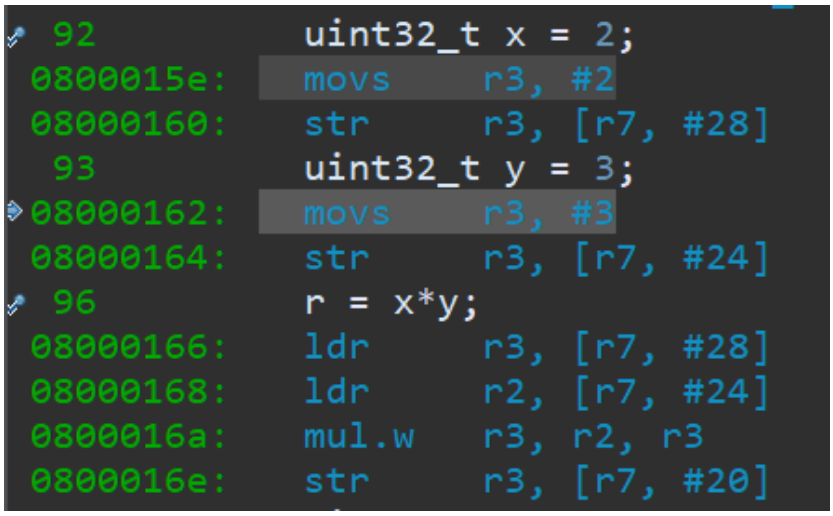
April 2025

1 Problema 1

Cree un proyecto en STM32CubeIDE para el microcontrolador STM32f103c8 y verifique los siguientes puntos observando la salida de disassembly (archivo .list o la vista de disassembly en la perspectiva de debug)

1.1 Inciso a

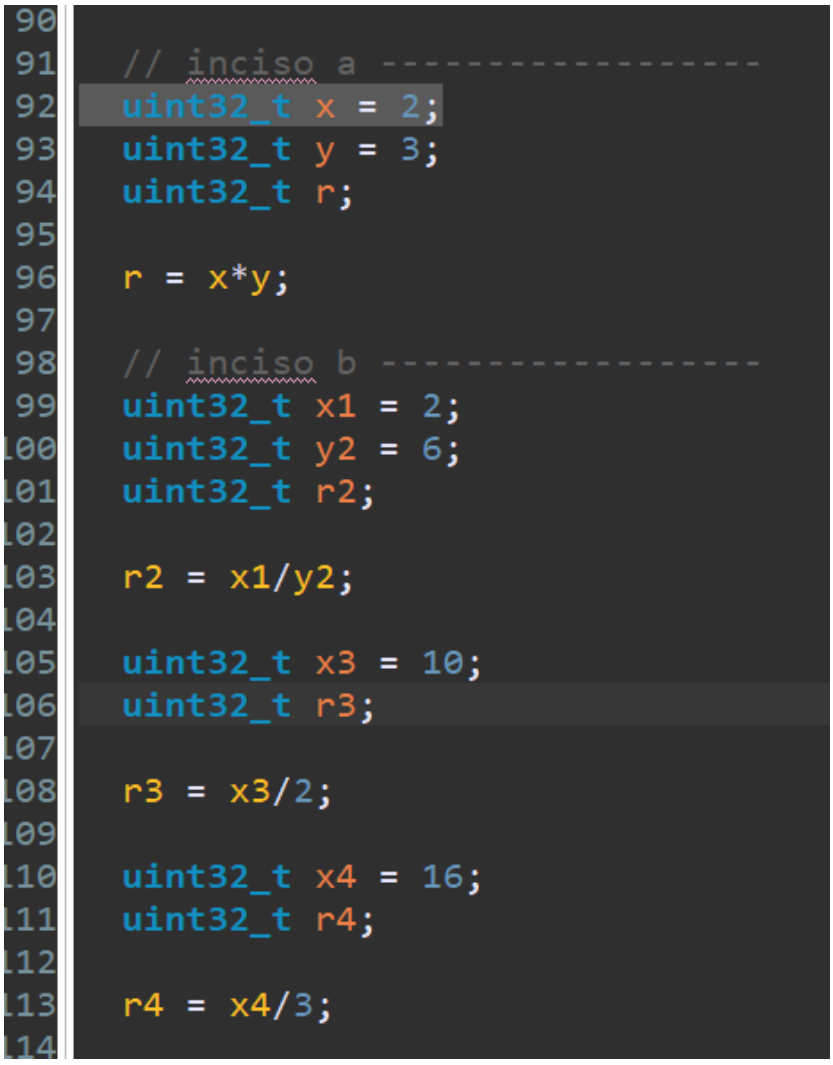
La implementación de la multiplicación en una instrucción de assembler, del tipo $r = x * y$ *donde las variables son* ***uint32_t*** *con valores dados en su declaración ¿se usa realmente una única instrucción para el proceso completo?*
No, el proceso de multiplicar lleva varias instrucciones de assembly. Hay que alojar los datos en los registros sin perder informacion, por lo que le lleva más operaciones



1.2 Inciso b

La implementación de la división de dos uint32_t en una instrucción. ¿Qué diferencia hay entre usar dos variables $r = x/y$ o usar una división constante como $r = x/128$ o $r = x/127$?

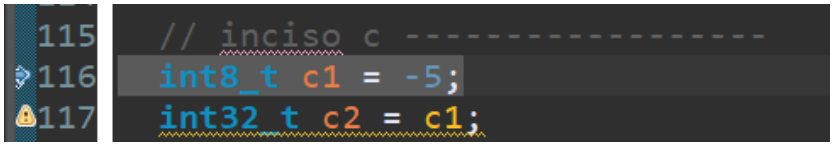
La variación es cuál instrucción, además de la cantidad que usa el compilador para realizar la operación, ya que cuando se usan dos variables se tiene un instrucción de assembly (*udiv*), pero en los casos donde hay una variable y una constante se tiene otra/otras instrucciones, porque hay veces que el compilador optimiza usando un shift register o "trucos" similares.

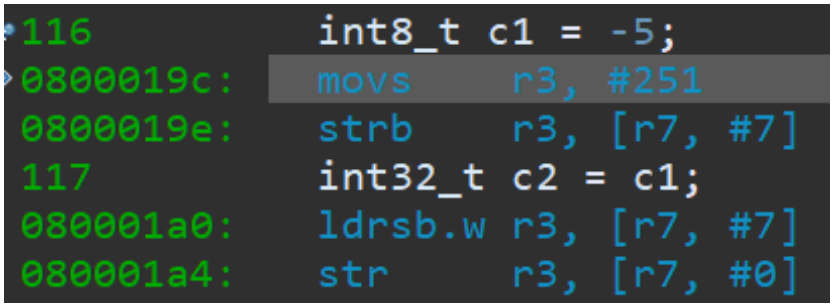


```
 92          uint32_t x = 2;
0800015e:    movs    r3, #2
08000160:    str     r3, [r7, #36]   @ 0x24
 93          uint32_t y = 3;
08000162:    movs    r3, #3
08000164:    str     r3, [r7, #32]
 96          r = x*y;
08000166:    ldr     r3, [r7, #36]   @ 0x24
08000168:    ldr     r2, [r7, #32]
0800016a:    mul.w   r3, r2, r3
0800016e:    str     r3, [r7, #28]
 99          uint32_t x1 = 2;
08000170:    movs    r3, #2
08000172:    str     r3, [r7, #24]
100          uint32_t y2 = 6;
08000174:    movs    r3, #6
08000176:    str     r3, [r7, #20]
103          r2 = x1/y2;
08000178:    ldr     r2, [r7, #24]
0800017a:    ldr     r3, [r7, #20]
0800017c:    udiv    r3, r2, r3
08000180:    str     r3, [r7, #16]
105          uint32_t x3 = 10;
08000182:    movs    r3, #10
08000184:    str     r3, [r7, #12]
108          r3 = x3/2;
08000186:    ldr     r3, [r7, #12]
08000188:    lsrs    r3, r3, #1
0800018a:    str     r3, [r7, #8]
110          uint32_t x4 = 16;
0800018c:    movs    r3, #16
0800018e:    str     r3, [r7, #4]
113          r4 = x4/3;
08000190:    ldr     r3, [r7, #4]
08000192:    ldr     r2, [pc, #12]   @ (0x80
08000194:    umull   r2, r3, r2, r3
08000198:    lsrs    r3, r3, #1
0800019a:    str     r3, [r7, #0]
```

1.3 Inciso c

La implementación del guardado de una variable int8_t con valor -5 en una variable int32_t





2 Problema 6

Si puede asignar la dirección de memoria correspondiente a cada registro, la siguiente función permite inicializar el periférico DWT y activar su función de conteo de ciclos para poder contar cuántos ciclos de clock se insumen durante la operación del microcontrolador.

```
void habilitar_contador_ciclos() {
    DEMCR |= (1 << 24);
    DWT_CTRL |= (1 << 0);
    DWT_CYCCNT = 0;
}
```

Rastree en la documentación, comenzando por el Reference Manual, el periférico DWT y los registros y bits utilizados.

2.1 Inciso a

¿A qué segmento del mapa de memoria pertenece? ¿Corresponde a ARM o a ST?

Pertenece al segmento de memoria de los periféricos del Cortex M3. Por lo que no es algo de ST, sino de la arquitectura Cortex M3

2.2 Inciso b

¿En qué documento se describe detalladamente la funcionalidad de cada registro?
Se puede encontrar la información en el manual de referencia (cortex_m3-RM.pdf)

3 Archivos

Se adjuntan junto al presente los archivos de proyecto de CubeIDE con el código con el que se resolvieron los problemas 1, 6 y 7. La definición de los registros se hizo en una librería denominada *GeneralPurposeLib.h*