Computer Systems 414 28 March 2021

Walt Deyzel: 21750793

Introduction:

The aim of assignment 2B is to write a C program using uVision5 that has a function that multiplies two variables and stores it in a static variable. The static variable has a specific place in memory that increments the value over time. The software allows us to see the assembly code alongside the c code. It also allows us to look at the contents any specific memory address.

Static int:

From the code below one can see that the contents of memory address **0x20000000** is loaded into register r0. Register r2 and register r1 is multiplied together and register r0 is added to the total before storing the value in r0. The memory address **0x20000000** is loaded into r3 and the calculated total is stored at that memory address for future use due to the static variable property.

```
0x000004E0 BF00      NOP
0x000004E2 E7FE      B        0x000004E2
     3: int mulc(int no1, int no2){
     4:
     5:          static int ans;
0x000004E4 4602      MOV      r2,r0
     6:          ans = (no1*no2) + ans;
0x000004E6 4803      LDR      r0,[pc,#12]   ; Address for static int: 0x20000000
0x000004E8 6800      LDR      r0,[r0,#0x00]
0x000004EA FB020001  MLA      r0,r2,r1,r0
0x000004EE 4B01      LDR      r3,[pc,#4]   ;
0x000004F0 6018      STR      r0,[r3,#0x00]
     7: }
```

Performance

Module/Function	Calls	Time(Sec)	Time(%)
sum		63.667 us	97%
main.c		61.583 us	94%
main	1	0.583 us	1%
mulc	1	1.167 us	2%
__main	1	0.250 us	0%
__scatterload_rt2	1	3.833 us	6%
__scatterload_copy	1	1.583 us	2%
__scatterload_zeroinit	1	54.167 us	82%
exit	0	0us	0%
__use_no_semihosting_swi	0	0us	0%
_sys_exit	0	0us	0%
RTE/Device/ARMCM3/system_ARMCM3.c		1.250 us	2%
RTE/Device/ARMCM3/startup_ARMCM3.s		0.833 us	1%

As you can see calling **mulc** 100 times takes up 44% of the total execution time. A total of 116 micro seconds vs only 1.167 micro seconds for calling mulc only once.

Module/Function	Calls	Time(Sec)	Time(%)
sum		262.583 us	99%
main.c		260.500 us	98%
main	1	84.000 us	32%
mulc	100	116.667 us	44%
__main	1	0.250 us	0%
__scatterload_rt2	1	3.833 us	1%
__scatterload_copy	1	1.583 us	1%
__scatterload_zeroinit	1	54.167 us	20%
exit	0	0us	0%
__use_no_semihosting_swi	0	0us	0%
_sys_exit	0	0us	0%
RTE/Device/ARMCM3/system_ARMCM3.c		1.250 us	0%
RTE/Device/ARMCM3/startup_ARMCM3.s		0.833 us	0%

The number of assembler code for the **main** function and **mulc** function is 14. As seen in the code snipper below.

Assembler code

```
                main:
0x000004D4 240F      MOVS     r4,#0x0F
0x000004D6 2511      MOVS     r5,#0x11
0x000004D8 4629      MOV      r1,r5
0x000004DA 4620      MOV      r0,r4
0x000004DC F000F802  BL.W     mulc (0x000004E4)
0x000004E0 BF00      NOP
0x000004E2 E7FE      B        0x000004E2
                mulc:
0x000004E4 4602      MOV      r2,r0
0x000004E6 4803      LDR      r0,[pc,#12]  ; @0x000004F4
0x000004E8 6800      LDR      r0,[r0,#0x00]
0x000004EA FB020001  MLA      r0,r2,r1,r0
0x000004EE 4B01      LDR      r3,[pc,#4]   ; @0x000004F4
0x000004F0 6018      STR      r0,[r3,#0x00]
0x000004F2 4770      BX       lr
```

Code: All code for assignment 2B.

```
1  int mulc(int no1, int no2);
2  int sum(int no1, int no2);
3
4  int sum(int no1, int no2){
5    int ans;
6      __asm{
7        add ans, no1, no2
8      }
9      return ans;
10 }
11
12 int mulc(int no1, int no2){
13
14   static int ans;
15   ans = (no1*no2) + ans;
16 }
17
18 int main(void){
19
20   int no1, no2;
21
22   no1 = 200;
23   no2 = 55;
24
25   for(int i = 0; i<100; i++)
26     mulc(no1, no2);
27
28   while(1);
29 }
```