Microprocessor and Assembly Language Lab 02

Kazi Shadman Sakib, FH-97

February 02, 2022

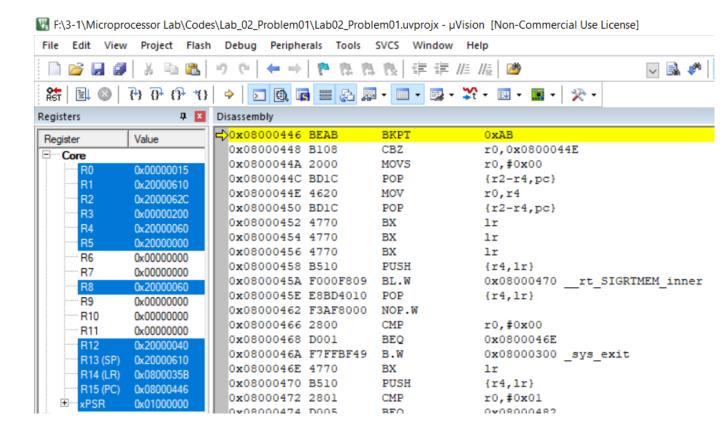
1 Task 01

1.1 Detailed Code Explanation

To solve this problem, I have first put the data in memory in the form of constants, where X = 9, Y = 8 and Z = 5 by using EQU (equate) Opcode. The equate directive (EQU) is used to substitute values for symbols or labels. Then in the main function, I have used the Opcode MOV to load the value of constants X, Y and Z in registers r4, r3 and r2 respectively, thus r4 = X, r3 = Y, r2 = Z. We simply then add the values in the registers r2 and r3, and store the result in register r0 by using the Opcode ADD. Lastly, we again use the Opcode ADD to add the values in the registers r0 and r4, and store the final result in the register r0.

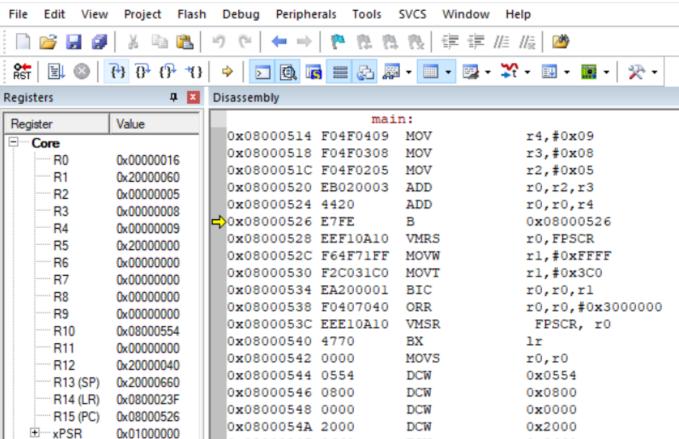
1.2 Screenshot of Debugger

1.2.1 After the Code has been Loaded



1.2.2 After the Code has been Executed

🔣 F:\3-1\Microprocessor Lab\Codes\Lab_02_Problem01\Lab02_Problem01.uvprojx - μVision [Non-Commercial Use Lice



2 Task 02

2.1 Detailed Code Explanation

To solve this problem, I have first created three variables X, Y and Z with their initial values as X = 9, Y = 8 and Z = 5 by using the Opcode DCD. The DCD directive allocates one or more words of memory, aligned on four-byte boundaries, and defines the initial run-time contents of the memory. After this, in the main function I have used the Opcode LDR to load the register r4, r3 and r2 with the content of memory location of X, Y and Z respectively, thus r4 = X, r3 = Y and r2 = Z. We simply then add the values in the registers r2 and r3, and store the result in register r0 by using the Opcode ADD. Lastly, we again use the Opcode ADD to add the values in the registers r0 and r4, and store the final result in the register r0.

2.2 Screenshot of Debugger

2.2.1 After the Code has been Loaded

🔣 F:\3-1\Microprocessor Lab\Codes\Lab 02 Problem02\Lab02 Problem02.uvprojx - μVision [Non-Commercial Use License] Project Flash Debug Peripherals Tools SVCS Window 10 (11 X 🗈 🏝 **←** ⇒ | (3) √(3) √(3) Registers ♣ Disassembly □ 0x08000446 BEAB BKPT 0xAB Register Value 0x08000448 B108 r0,0x0800044E CBZ Core 0x0800044A 2000 MOVS r0,#0x00 0x00000015 0x0800044C BD1C POP {r2-r4,pc} 0x20000610 0x0800044E 4620 MOV r0,r4 R2 0x2000062C 0x08000450 BD1C POP {r2-r4,pc} R3 0x00000200 0x08000452 4770 1 m R4 0x20000060 0x08000454 4770 lr 0x20000000 0x08000456 4770 BXlr R6 0x00000000 0x08000458 B510 PUSH {r4.lr} R7 0x00000000 0x0800045A F000F809 BL.W 0x08000470 rt SIGRTMEM inner R8 0x0800045E E8BD4010 POP {r4, lr} R9 0x00000000 0x08000462 F3AF8000 NOP.W 0x00000000 R10 0x08000466 2800 CMP r0,#0x00 0x00000000 R11 0x08000468 D001 BEQ 0x0800046E 0x20000040 0x0800046A F7FFBF49 B.W 0x08000300 _sys_exit 0x20000610 R13 (SP) 0x0800046E 4770 BX 1r R14 (LR) 0x0800035B 0x08000470 B510 PUSH {r4, lr} R15 (PC) 0x08000446 0x08000472 2801 CMP r0,#0x01 xPSR 0x01000000 0.000000482

2.2.2 After the Code has been Executed

🕎 F:\3-1\Microprocessor Lab\Codes\Lab_02_Problem02\Lab02_Problem02.uvprojx - μVision [Non-Commercial Use License] Project Flash Debug Peripherals Tools SVCS Window Help X 🖹 🖺 🗸 Registers ♣ Disassembly main: Register Value 0x08000514 4C03 T.DR r4, [pc, #12] ; @0x08000524 - Core 0x08000516 4B04 T.DR r3, [pc, #16] ; @0x08000528 R₀ 0x00000016 0x08000518 4A04 r2,[pc,#16] ; @0x0800052C LDR R1 0x20000060 0x0800051A EB020003 ADD r0, r2, r3 R2 0x00000005 0x0800051E 4420 ADD r0, r0, r4 R3 0x00000008 0x08000520 E7FE 0x08000520 В R4 0x00000009 0x08000522 0000 DCW 0×00000 R5 0x20000000 0x08000524 0009 0x0009 DCW R6 0x00000000 0x08000526 0000 DCW 0×00000 R7 0x00000000 0x08000528 0008 DCW 0x0008 R8 0x00000000 0x0800052A 0000 DCW 0×00000 R9 0x00000000 0x0800052C 0005 DCW 0×00005 R10 0x0800055C 0x0800052E 0000 DC:W 0x0000R11 0x00000000 0x08000530 EEF10A10 VMRS r0.FPSCR 0x20000040 0x08000534 F64F71FF MOVW rl.#0xFFFF R13 (SP) 0x20000660 0x08000538 F2C031C0 r1,#0x3C0 MOVT R14 (LR) 0x0800023F 0x0800053C EA200001 r0, r0, r1 BIC R15 (PC) 0x08000520 0x08000540 F0407040 ORR r0,r0,#0x3000000 ± ·····xPSR 0x01000000 0.000000544 FFF10710 TRICE EDSCD vo

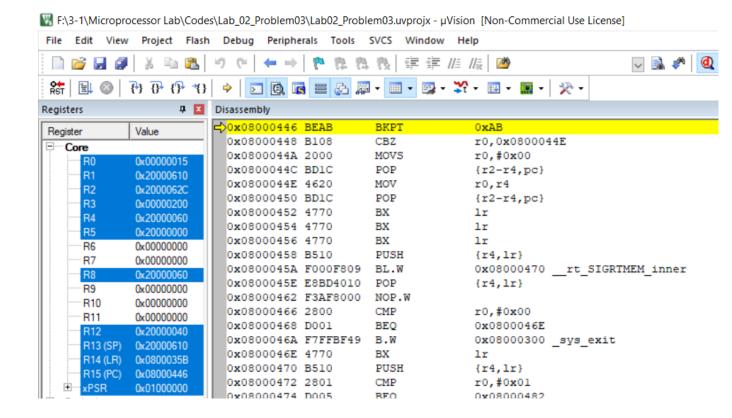
3 Task 03

3.1 Detailed Code Explanation

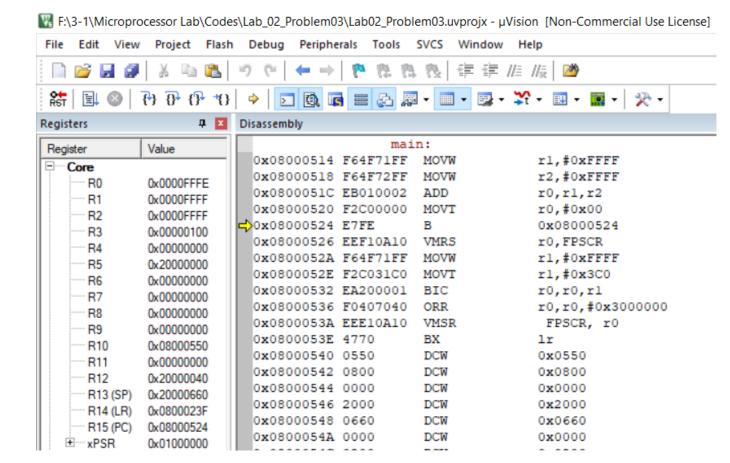
To solve this problem, I have first used the Opcode MOVW to load only 16 bits [15:0] to the registers r1 and r2 with the hexa-decimal values 0xFFFF. Then I have used the Opcode ADD to find the addition of the two 16 bit variables. Lastly, I have used the Opcode MOVT to load 0's in the upper 16 bit [31:16] of the final result, which is in register r0. Opcode MOVT writes to Rd[31:16], without affecting Rd[15:0]. Thus the required result is in the register r0.

3.2 Screenshot of Debugger

3.2.1 After the Code has been Loaded



3.2.2 After the Code has been Executed



4 Task 04

4.1 Detailed Code Explanation

To solve this problem, I have first created two integer variables value1 and value2, and assigned their initial values as value1 = 8 and value2 = 9 by using the Opcode DCD. The DCD directive allocates one or more words of memory, aligned on four-byte boundaries, and defines the initial run-time contents of the memory. After this, in the main function I have used the Opcode LDR to load the register r1 and r2 with the content of memory location of value1 and value2 respectively, thus r1 = value1 and r2 = value2. To compare the values of these two registers I have used the Opcode CMP. The Opcode CMP compares the two operands. The BLT (branch less than) instruction is one of several conditional branch instructions. The conditional transfer instruction BLT transfers control to the statement labeled "done" if the contents of r1 contains the smaller integer number. Otherwise, the next instruction is executed. At "done", register r0 will always contain the smaller of the two integer numbers as the final result.

4.2 Screenshot of Debugger

4.2.1 After the Code has been Loaded

🕎 F:\3-1\Microprocessor Lab\Codes\Lab 02 Problem04\Lab02 Problem04.uvproix - μVision [Non-Commercial Use License] File Edit View Project Flash Debug Peripherals Tools SVCS Window Help 🗸 🔒 🦚 🛛 🗨 **않** 🗐 🚳 | 한 만 안 *8 | 💠 | 🖸 👰 🕞 🗏 👰 🛒 * 📖 * 🏋 * 🔟 * 🛒 * 🏃 * 4 Disassembly Registers □>0x08000446 BEAB 0xAB BKPT Register 0x08000448 B108 CBZ r0,0x0800044E Core 0x0800044A 2000 MOVS r0.#0x00 0x00000015 0x0800044C BD1C {r2-r4,pc} POP 0x20000610 0x0800044E 4620 MOV r0.r4 0x2000062C 0x08000450 BD1C POP {r2-r4,pc} 0x00000200 0x08000452 4770 1 r R4 0x20000060 0x08000454 4770 BXR5 0x20000000 0x08000456 4770 BX 1r R6 0x00000000 0x08000458 B510 PUSH {r4,lr} R7 0x00000000 0x0800045A F000F809 BL.W 0x08000470 __rt_SIGRTMEM_inner 0x20000060 0x0800045E E8BD4010 POP {r4,lr} 0x00000000 0x08000462 F3AF8000 NOP.W R10 0×000000000 0x08000466 2800 CMP r0.#0x00 R11 0x00000000 0x08000468 D001 BEQ 0x0800046E 0x20000040 0x0800046A F7FFBF49 B.W 0x08000300 sys_exit 0x20000610 R13 (SP) 0x0800046E 4770 BX 1r R14 (LR) 0x0800035B 0x08000470 B510 PUSH {r4.lr} 0x08000446 R15 (PC) 0x08000472 2801 CMP r0,#0x01 xPSR 0x01000000 n⊎nennn474 bnns 0.000000482 BEO.

4.2.2 After the Code has been Executed

🔣 F:\3-1\Microprocessor Lab\Codes\Lab_02_Problem04\Lab02_Problem04.uvprojx - µVision [Non-Commercial Use License] File Edit View Project Flash Debug Peripherals Tools SVCS Window Help v 🔒 🦚 10 (21 ← → | № 性 性 後 | 様 | 様 | Ø X 📭 🖺 ** | 🗐 🚳 | 한 안 안 *8 | 💠 | 🖸 📵 📭 🚍 📮 - 🔟 - 👺 - 🏋 - 🔟 - 🌉 - 🎠 -4 Z Disassembly Registers main: Register Value 0x08000514 4903 rl, [pc, #12] ; @0x08000524 LDR Core 0x08000516 4A04 LDR r2, [pc, #16] ; @0x08000528 R0 0x00000008 0x08000518 4291 CMP r1, r2 R1 0x00000008 0x0800051A DB00 BLT 0x0800051E R2 0x00000009 0x0800051C 4611 MOV r1, r2 R3 0x00000100 0x0800051E 4608 MOV r0,rl R4 0x00000000 □ 0x08000520 E7FE 0x08000520 В R5 0x20000000 0x08000522 0000 DCW 0x0000 R6 0x00000000 0x08000524 0008 DCW 0x0008 · R7 0x0000000000x08000526 0000 DCW 0x0000-R8 0x00000000 0x08000528 0009 DCW 0x0009R9 0x00000000 0x0800052A 0000 DCW 0x0000 R10 0x08000558 0x0800052C EEF10A10 VMRS r0, FPSCR R11 0x00000000 0x08000530 F64F71FF MOVW rl.#0xFFFF 0x20000040 0x08000534 F2C031C0 MOVT rl,#0x3C0 0x20000660 R13 (SP) 0x08000538 EA200001 BIC r0,r0,r1 R14 (LR) 0x0800023F 0x0800053C F0407040 ORR r0,r0,#0x3000000 R15 (PC) 0x08000520 0x08000540 EEE10A10 VMSR FPSCR, r0 xPSR 0x81000000 0.000000544 4770