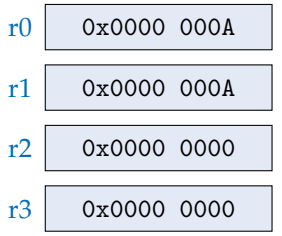
Homework 4: Answers

Profs. John McLeod & Arash Reyhani

ECE3375, Winter 2022

Problem: Some registers and memory cells in a microcontroller are initialized as shown. The following code is executed:

```
1 adds r2, r1, r0
2 subeq r2, r1
3 addne r2, r1
4 cmp r2, r0
5 moveq r3, r1, lsl #2
6 movne r3, r2, lsl #2
7 subs r0, r1, r3
8 subvs r2, r3
```



What are the contents of the registers after this code is executed?

The status flags are updated after executing lines 1, 4, and 7. What are they?

Solution: Trace the code line-by-line.

1. Register `r2` now holds the value $(10)_{10} + (10)_{10} = (20)_{10} =$ `0x14`. This is ordinary addition of small, positive numbers, so there is no carry-out, no overflow, and the result is neither negative nor zero. The flags remain `NZCV = 0000`.

2. The `eq` comparison flag checks if `Z = 1`. This is not true, so this line does not execute.

3. The `ne` comparison flag checks if `Z = 0`. This is true, so `r2` now hods the value $(20)_{10}$ from previous $+(10)_{10} = (30)_{10} =$ `0x1E`.

4. The value in `r2` is greater than the value in `r1`. As `cmp` performs a subtraction using 2's compliment, there is a carry-out (meaning, there is no borrow — the subtraction was successful). There is no overflow, and the answer is neither negative nor zero, so `NZCV = 0010`.

5. As value in `r2` is not equal to the value in `r1`, this line does not execute.

6. As value in `r2` is not equal to the value in `r1`, this line does execute. A `lsl #2` is equivalent to multiplying the decimal value by $\times 4$, so `r3` now holds $4 \times (30)_{10} = (120)_{10} =$ `0x78`.

7. The subtraction is $(10)_{10} - (120)_{10} = (-110)_{10}$, expressed in 2's complement register `r0` holds the value $(-110)_{10} =$

`0xFFFFFF92`. This is negative, and would have required a borrow if it was unsigned subtraction (so the carry-out is zero). The flags are `NZCV = 1000`.

8. The `vs` ("overflow set") comparison flag checks if `V = 1`. This is not true, so this line does not execute.

Final register contents:

r0	`0xFFFF FF92`
r1	`0x0000 000A`
r1	`0x0000 001E`
r1	`0x0000 0078`