Assignment - 2

TT0L - GROUP 0

Person 1	1111111111
Person 2	1111111111

(Person 1 ID + Person 2 ID) % 30 = 25

Q25. The 32 bits value stored in 0x2000 till 0x2000 + 36 is actually the timestamp in BCD format. For example, 00:HH:MM:SEC. For example 0x00233435 is represented as 23 hour 34 min 35 second. If the time format is valid, store "1111" in 0x2100 till (0x2100 + 36). Else store "2222" in 0x2100 till (0x2100 + 36).

Initialized data and stored it in 0x2000 - 0x2024 (0x2000 + 36)

0x11111111 (0x2000)

0x22223333 (0x2004)

0x31111111 (0x2008)

0x42223333 (0x200C)

0x51111111 (0x2010)

0x62223333 (0x2014)

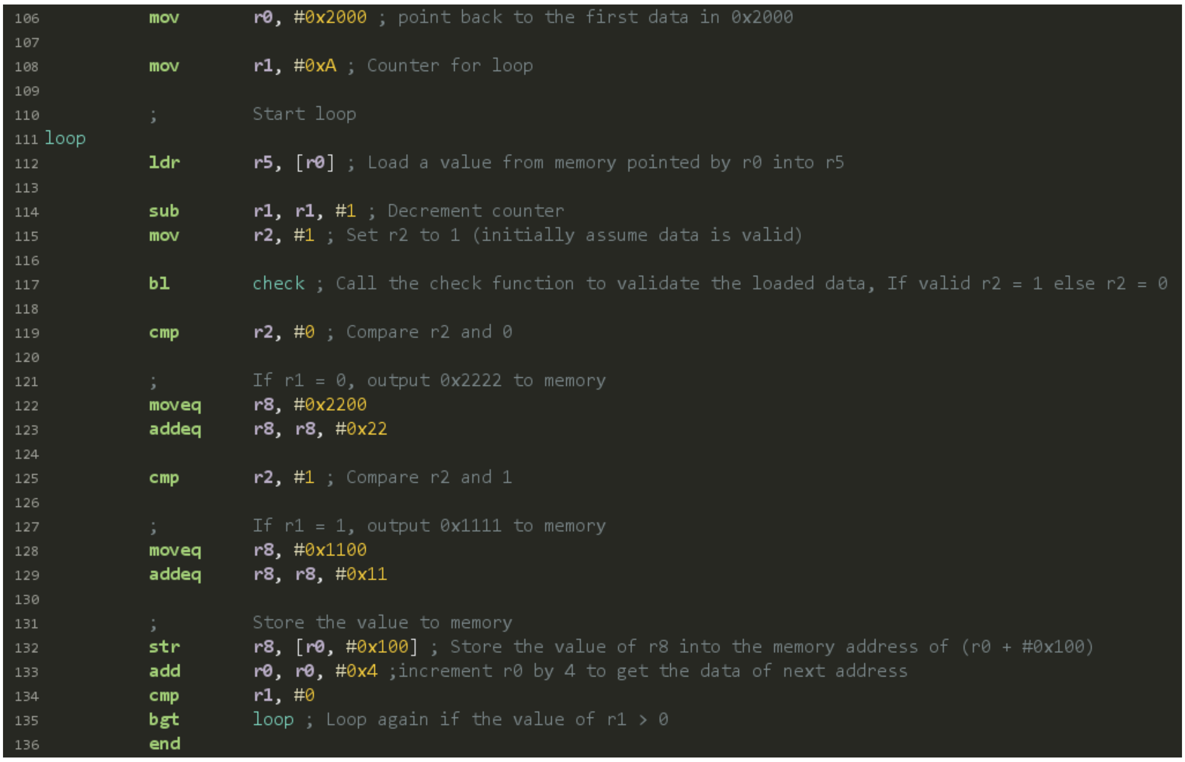
0x71111111 (0x2018)

0x82000000 (0x201C)

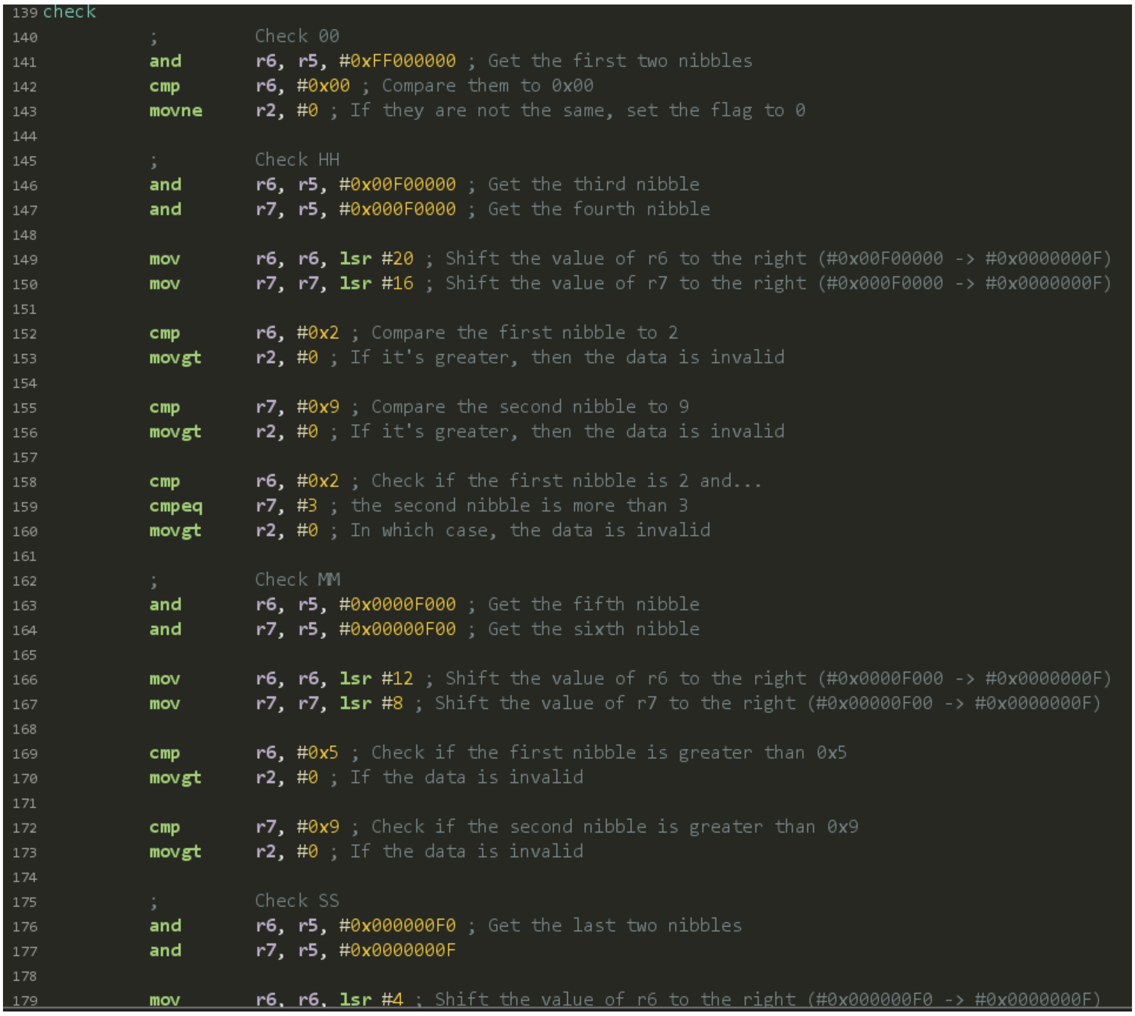
0x91111111 (0x2020)

0xA2223333 (0x2024)

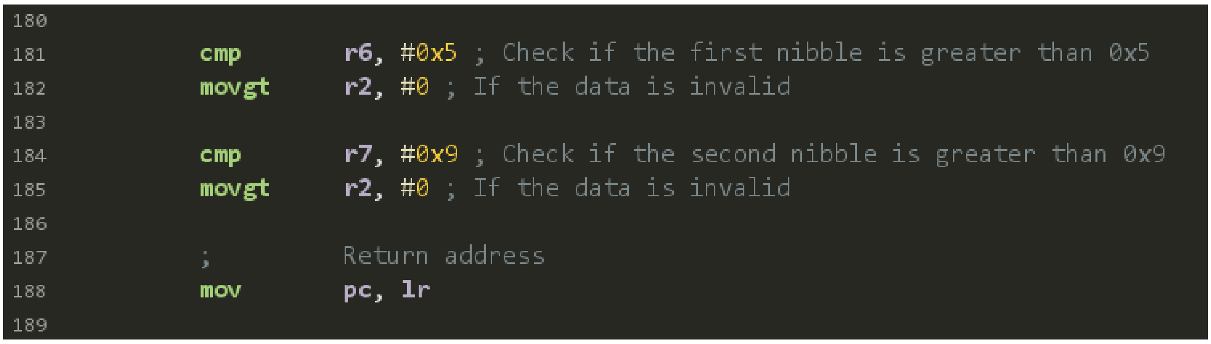
Program Snapshot



```
        mov     r0, #0x2000 ; point back to the first data in 0x2000

        mov     r1, #0xA ; Counter for loop

        ;       Start loop
loop
        ldr     r5, [r0] ; Load a value from memory pointed by r0 into r5

        sub     r1, r1, #1 ; Decrement counter
        mov     r2, #1 ; Set r2 to 1 (initially assume data is valid)

        bl      check ; Call the check function to validate the loaded data, If valid r2 = 1 else r2 = 0

        cmp     r2, #0 ; Compare r2 and 0

        ;       If r1 = 0, output 0x2222 to memory
        moveq   r8, #0x2200
        addeq   r8, r8, #0x22

        cmp     r2, #1 ; Compare r2 and 1

        ;       If r1 = 1, output 0x1111 to memory
        moveq   r8, #0x1100
        addeq   r8, r8, #0x11

        ;       Store the value to memory
        str     r8, [r0, #0x100] ; Store the value of r8 into the memory address of (r0 + #0x100)
        add     r0, r0, #0x4 ;increment r0 by 4 to get the data of next address
        cmp     r1, #0
        bgt     loop ; Loop again if the value of r1 > 0
        end
```

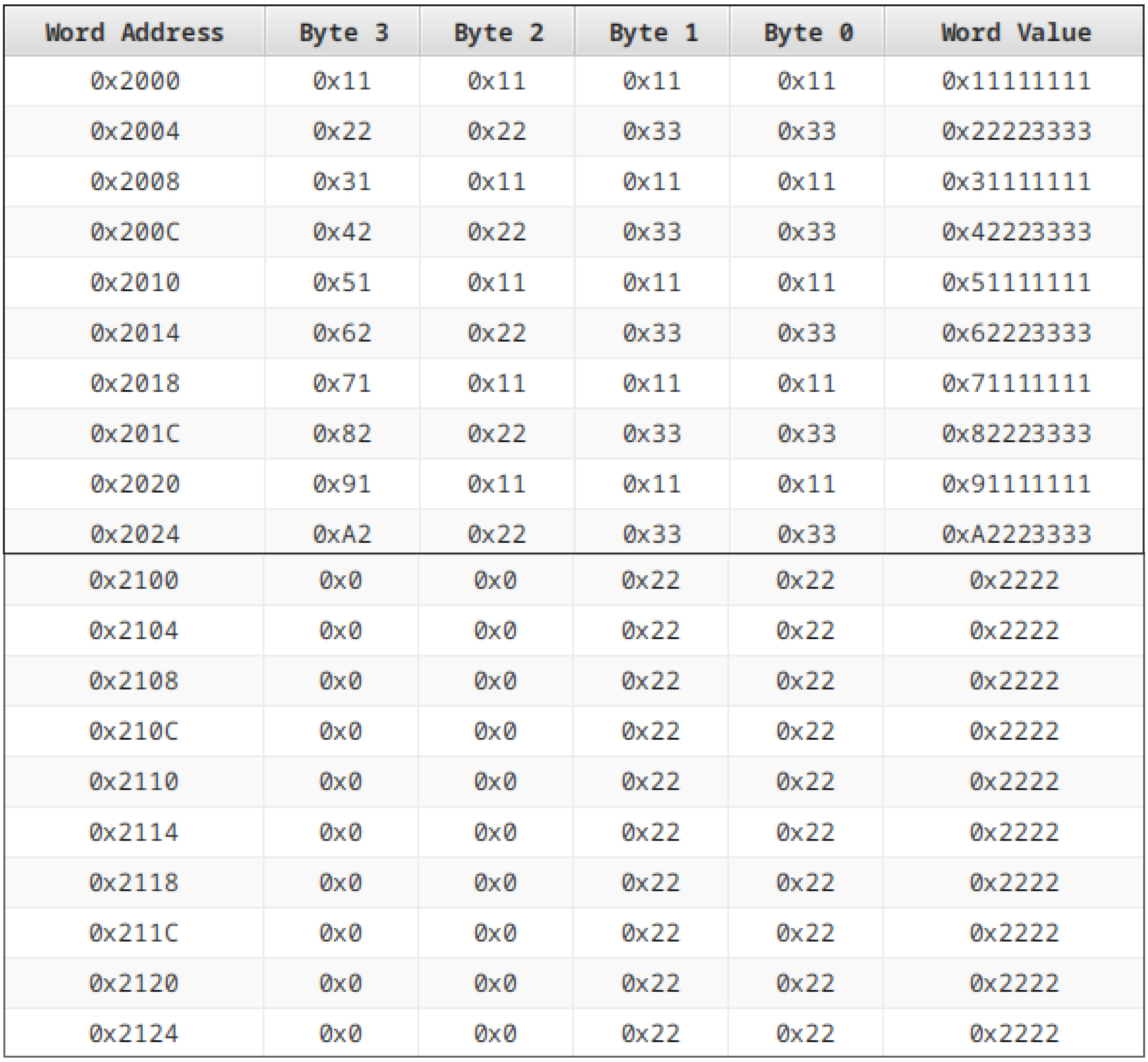


```
check
        ;       Check 00
        and     r6, r5, #0xFF000000 ; Get the first two nibbles
        cmp     r6, #0x00 ; Compare them to 0x00
        movne   r2, #0 ; If they are not the same, set the flag to 0

        ;       Check HH
        and     r6, r5, #0x00F00000 ; Get the third nibble
        and     r7, r5, #0x000F0000 ; Get the fourth nibble

        mov     r6, r6, lsr #20 ; Shift the value of r6 to the right (#0x00F00000 -> #0x0000000F)
        mov     r7, r7, lsr #16 ; Shift the value of r7 to the right (#0x000F0000 -> #0x0000000F)

        cmp     r6, #0x2 ; Compare the first nibble to 2
        movgt   r2, #0 ; If it's greater, then the data is invalid

        cmp     r7, #0x9 ; Compare the second nibble to 9
        movgt   r2, #0 ; If it's greater, then the data is invalid

        cmp     r6, #0x2 ; Check if the first nibble is 2 and...
        cmpeq   r7, #3 ; the second nibble is more than 3
        movgt   r2, #0 ; In which case, the data is invalid

        ;       Check MM
        and     r6, r5, #0x0000F000 ; Get the fifth nibble
        and     r7, r5, #0x00000F00 ; Get the sixth nibble

        mov     r6, r6, lsr #12 ; Shift the value of r6 to the right (#0x0000F000 -> #0x0000000F)
        mov     r7, r7, lsr #8 ; Shift the value of r7 to the right (#0x00000F00 -> #0x0000000F)

        cmp     r6, #0x5 ; Check if the first nibble is greater than 0x5
        movgt   r2, #0 ; If the data is invalid

        cmp     r7, #0x9 ; Check if the second nibble is greater than 0x9
        movgt   r2, #0 ; If the data is invalid

        ;       Check SS
        and     r6, r5, #0x000000F0 ; Get the last two nibbles
        and     r7, r5, #0x0000000F

        mov     r6, r6, lsr #4 ; Shift the value of r6 to the right (#0x000000F0 -> #0x0000000F)
```



```

        cmp     r6, #0x5 ; Check if the first nibble is greater than 0x5
        movgt   r2, #0 ; If the data is invalid

        cmp     r7, #0x9 ; Check if the second nibble is greater than 0x9
        movgt   r2, #0 ; If the data is invalid

        ;       Return address
        mov     pc, lr

```

Snapshot of memory address



Word Address	Byte 3	Byte 2	Byte 1	Byte 0	Word Value
0x2000	0x11	0x11	0x11	0x11	0x11111111
0x2004	0x22	0x22	0x33	0x33	0x22223333
0x2008	0x31	0x11	0x11	0x11	0x31111111
0x200C	0x42	0x22	0x33	0x33	0x42223333
0x2010	0x51	0x11	0x11	0x11	0x51111111
0x2014	0x62	0x22	0x33	0x33	0x62223333
0x2018	0x71	0x11	0x11	0x11	0x71111111
0x201C	0x82	0x22	0x33	0x33	0x82223333
0x2020	0x91	0x11	0x11	0x11	0x91111111
0x2024	0xA2	0x22	0x33	0x33	0xA2223333
0x2100	0x0	0x0	0x22	0x22	0x2222
0x2104	0x0	0x0	0x22	0x22	0x2222
0x2108	0x0	0x0	0x22	0x22	0x2222
0x210C	0x0	0x0	0x22	0x22	0x2222
0x2110	0x0	0x0	0x22	0x22	0x2222
0x2114	0x0	0x0	0x22	0x22	0x2222
0x2118	0x0	0x0	0x22	0x22	0x2222
0x211C	0x0	0x0	0x22	0x22	0x2222
0x2120	0x0	0x0	0x22	0x22	0x2222
0x2124	0x0	0x0	0x22	0x22	0x2222

All the initialized time format are invalid as the first two nibbles of all the values are not equal to 00

Modified initialized data with the test data given in the instructions

0x11111111 -> 0x00000001

0x22223333 -> 0x00990011

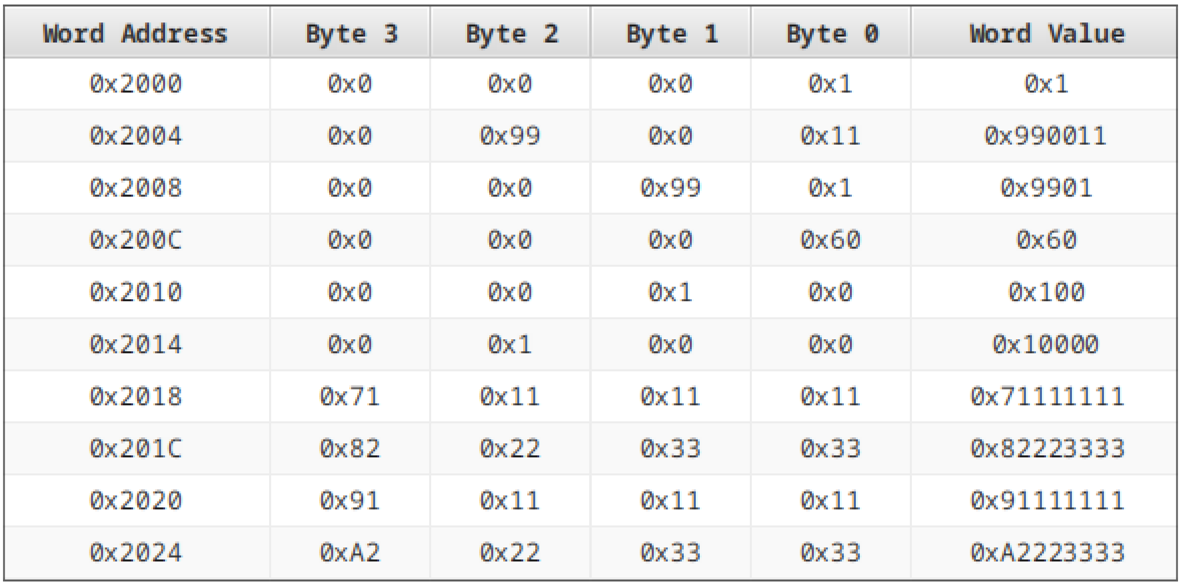
0x31111111 -> 0x00009901

0x42223333 -> 0x00000060

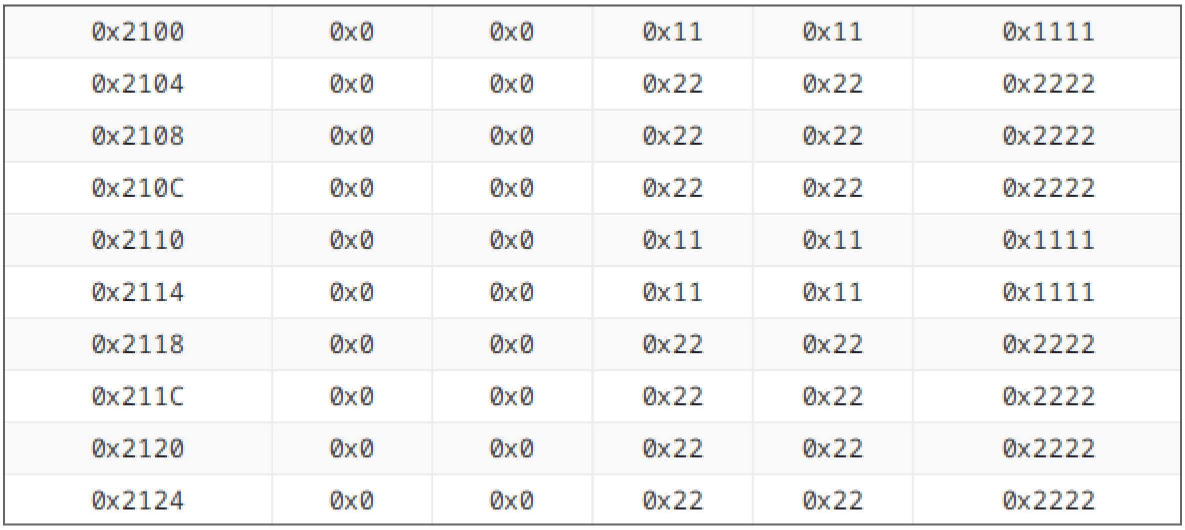
0x51111111 -> 0x00000100

0x62223333 -> 0x00010000

Data from 0x2018 - 0x2024 will remain unchanged.



Word Address	Byte 3	Byte 2	Byte 1	Byte 0	Word Value
0x2000	0x0	0x0	0x0	0x1	0x1
0x2004	0x0	0x99	0x0	0x11	0x990011
0x2008	0x0	0x0	0x99	0x1	0x9901
0x200C	0x0	0x0	0x0	0x60	0x60
0x2010	0x0	0x0	0x1	0x0	0x100
0x2014	0x0	0x1	0x0	0x0	0x10000
0x2018	0x71	0x11	0x11	0x11	0x71111111
0x201C	0x82	0x22	0x33	0x33	0x82223333
0x2020	0x91	0x11	0x11	0x11	0x91111111
0x2024	0xA2	0x22	0x33	0x33	0xA2223333



0x2100	0x0	0x0	0x11	0x11	0x1111
0x2104	0x0	0x0	0x22	0x22	0x2222
0x2108	0x0	0x0	0x22	0x22	0x2222
0x210C	0x0	0x0	0x22	0x22	0x2222
0x2110	0x0	0x0	0x11	0x11	0x1111
0x2114	0x0	0x0	0x11	0x11	0x1111
0x2118	0x0	0x0	0x22	0x22	0x2222
0x211C	0x0	0x0	0x22	0x22	0x2222
0x2120	0x0	0x0	0x22	0x22	0x2222
0x2124	0x0	0x0	0x22	0x22	0x2222

The value in 0x2100, 0x2110 and 0x2114 is valid.