
Lab 6. Efficient Load and Store in Assembly

1. Tasks of the lab

We have practiced in Workshop 3 about how to use load/store instructions in various forms to load and save 8-, 16-, and 32-bit signed and unsigned data from and to the memory. In that project, we used global variables to pass parameters in order to have a better understanding of the pseudo LDR instruction. In this lab, we modify the assembly functions of Workshop 3 so that we do not need to rely on global variables to pass the parameters. To this end, we need to redefine the prototype of the assembly functions as

```
extern void task10(int8_t *pInt8, int8_t num);
extern void task11(uint8_t *pUInt8, int8_t num);
extern void task12(int8_t *pInt8, int16_t *pInt16, int8_t num);
extern void task13(uint8_t *pUInt8, uint16_t *pUInt16, int8_t num);
extern void task14(int16_t *pInt16, int32_t *pInt32, int8_t num);
extern void task15(uint16_t *pUInt16, uint32_t *pUInt32, int8_t num);
```

To give you an example of the revision of the assembly code, the modified code for Task 12 is provided below (see the enclosed code for Workshop 3 for the original code)

```
task12 PROC
    PUSH {r4-r5, lr}
    ; r0 = gPtrArray10a
    ; r1 = gPtrArray12a
    ; r2 = gVar1
    MOV r3, #0
task12_loop
    CMP r3, r2
    BGE task12_end
    LDRSB r4, [r0, r3]
    LDR r5, =10
    SUB r5, r4
    STRH r5, [r1, r3, LSL #1]
    ADD r4, #1
    STRB r4, [r0, r3]
    ADD r3, #1
    B task12_loop
task12_end
    POP {r4-r5, pc}
ENDP
```

Your work for this lab is to finish the revision of the other 5 tasks of Workshop 3, the points for each task is listed below:

File failed to load: <https://www.erau.us/wiki/xypic.js>

- Task 10: 10 points since the original code is given.

-
- Task 11: 20 points since the original code is missing from the provided code.
 - Task 13: 30 points since the original code is missing from the provided code.
 - Task 14: 10 points since the original code is given.
 - Task 15: 30 points since the original code is missing from the provided code.

2. Lab report

Like Lab 5, this lab report is special as well—the results of Tasks 11, 13, and 15 will be used to assess an ABET program outcome. The grading will be following the rubrics in the enclosed pdf file. Please read it carefully before doing your programming.

You need to add appropriate comments to all your code. Copy all the code snippets you changed/modified together with your comments to your report. Please be reminded that you need to perform your programming as independently from the TA as possible as seeking too much help will lead to unsatisfactory of some Attributes. Note that each of Tasks 11, 13, and 15 will be assessed using Attributes a, b, c, and d.

```

0x20010000: -015 -011 -007 -003 001 005 009 013 000 000 000 000 000 000 000 000
0x20010010: 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
0x20010020: -015 -011 -007 -003 001 005 009 013 000 000 000 000 000 000 000 000
0x20010030: 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000

```

Fig. 1. Results for Task 10. Display parameters: decimal, signed, char.

```

0x20010040: 002 034 066 098 130 162 194 226 000 000 000 000 000 000 000 000
0x20010050: 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
0x20010060: 002 034 066 098 130 162 194 226 000 000 000 000 000 000 000 000
0x20010070: 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000

```

Fig. 2. Results for Task 11. Display parameters: decimal, unsigned, char.

```

0x20010000: -014 -010 -006 -002 002 006 010 014 000 000 000 000 000 000 000 000
0x20010010: 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
0x20010020: -014 -010 -006 -002 002 006 010 014 000 000 000 000 000 000 000 000
0x20010030: 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000

```

Fig. 3. Results for Task 12. Display parameters: decimal, signed, char. Note that the results here are the increment results of Fig. 1

```

0x20010080: 00025 00021 00017 00013 00009 00005 00001 -00003
0x20010090: 00000 00000 00000 00000 00000 00000 00000 00000
0x200100A0: 00025 00021 00017 00013 00009 00005 00001 -00003
0x200100B0: 00000 00000 00000 00000 00000 00000 00000 00000

```

Fig. 4. Results for Task 12. Display parameters: decimal, signed, short.

```

0x200100C0: 00036 00100 00164 00228 00292 00356 00420 00000
0x200100D0: 00000 00000 00000 00000 00000 00000 00000 00000
0x200100E0: 00036 00100 00164 00228 00292 00356 00420 00000
0x200100F0: 00000 00000 00000 00000 00000 00000 00000 00000

```

Fig. 5. Results for Task 13. Display parameters: decimal, unsigned, short.

```

0x20010100: 0000000193 0000000157 0000000121 0000000085
0x20010110: 0000000049 0000000013 -0000000023 0000000000
0x20010120: 0000000193 0000000157 0000000121 0000000085
0x20010130: 0000000049 0000000013 -0000000023 0000000000

```

Fig. 6. Results for Task 14. Display parameters: decimal, signed, int.

```

0x20010140: 0000001636 0000002724 0000003812 0000004900
0x20010150: 0000005988 0000007076 000000420 0000000000
0x20010160: 0000001636 0000002724 0000003812 0000004900
0x20010170: 0000005988 0000007076 000000420 0000000000

```

Fig. 7. Results for Task 15. Display parameters: decimal, unsigned, int.