CSE/EE 5/7385 Microcontroller Architecture and Interfacing

HOMEWORK 2

1. An embedded systems engineer is examining the machine code for a store instruction with a pre-indexed operand. The hexadecimal code is:

`0xE583706C`

The general form of the store instruction is:

`STR <Rd>, [<Rn>, #<immed_5>*4]`

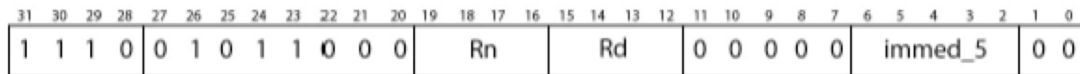
a) Give the actual destination register: ___**r7**____

b) Give the actual base address register: ___**r3**_____

c) Give the actual address offset value (in hex): __**0x6C**__

Slide 16 in week3 (ARM-Load-Store) gives the following format:

STR <Rd>, [<Rn>, #<immed_5> * 4]



32 bit ARM STR

Separating the hex into a 32-bit string:

1110-01011000-0011-0111-00000-11011-00

So <Rn> is R3 due to the (0011) string and <Rd> is R7 due to the (0111) string. The immediate address offset is given by (11011-00) which is 0x6C.

2. Assume that the memory of an ARM processor has the following content and that the ARM processor is operating with little endian access:

ADDRESS	DATA
0x4010	20
0x4011	40
0x4012	00
0x4013	00
0x4014	DA
0x4015	14
0x4016	40
0x4017	00
0x4018	00
0x4019	00
0x401A	40
0x401B	1C
0x401C	1C
0x401D	1C
0x401E	40
0x401F	00
0x4020	00

Give the content of the destination register after the following instructions execute.

a)
```
ldr   r0, =0x4018        ;r0 = _____0x00004018______
```

b)
```
ldr   r1, =0x4018
ldr   r3, [r1]           ;r3 = ___0x1C400000_________
```

c)
```
ldr   r1, =0x4018
ldr   r3, [r1, #4]       ;r3 = ___0x00401C1C_________
```

d)
```
ldr   r1, =0x4014
ldrsh r3, [r1]           ;r3 = ___0x000014DA_________
```

e)
```
ldr   r1, =0x4014
ldrsb r3, [r1]           ;r3 = ___0xFFFFFFDA_________
```

g)
```
ldr   r1, =0x4014
ldrsb r3, [r1, #4]       ;r3 = _0x00000000_, r1 =_0x0004014_
```

h)
```
ldr   r1, =0x4014
ldrsb r3, [r1, #4]!      ;r3 = _0x00000000_, r1 =_0x0004018_
```

i)
```
ldr   r1, =0x4010
ldr   r2, =0x3
ldr   r3, [r1, r2, lsl #2]!    ;r3 = _0x000014DA_, r1 =_0x000401C
```

3. An embedded systems designer is implementing a stack. She must decide which of the following multiple register load instructions should be used for the "pop" and "push" instructions. There are four possible types of stacks listed below. Give the appropriate form of **ldm***xx* and **stm***xx* instructions to use for each stack type where *xx* may be **ia**, **ib**, **da**, or **db**.

Fully Ascending Stack: "pop" _**ldmda**___ "push" __**stmib**_________

Fully Descending Stack: "pop" _**ldmia**__ "push" __**stmdb**________

Empty Ascending Stack: "pop" __**ldmdb**_ "push" ____**stmia**_______

Empty Descending Stack: "pop" __**ldmib**__ "push" __**stmda**______

4. Consider the following ARM program segment and assume that the 32-bit data bus requires 2 clock cycles for the transfer of a single word. The microcontroller system clock is running at 100 MHz. Give the data transfer bandwidth in Mb/s (Megabits per second). NOTE: Mb is MegaBITS NOT MegaBYTES.

```
                ;r0 points to the start of the source data
                ;r3 points to the start of the destination data

                mov    r2, #0x1             ;r2<--1
                mov    r4, #0x1, 30         ;r4<-- 1*2^2 = 4
looplab         ldmia  r0!,   {r5-r7}       ;load 24 bytes (24*2=48 clock cycles)
                stmia  r3!,   (r5-r7}       ;store 24 bytes (24*2=48 clock cycles)
                cmp    r4, r2               ;subtract r4-r2 to set flags (r4-1)
                mov    r4, r4, 1            ;shift r4 one bit to the right (4,2,1)
                bne    looplab              ;branch if Z=1
stop            b      stop
```

Bandwidth in Mb/s = __________400_____

Each time through the loop there are 24 bytes loaded and 24 bytes stored, so there are 48 bus transactions and there are 96 clock cycles. The loop is executed three times since first time r4=4 is compared to r2=1, second time r4=2 is compared to r2=1, and third time r4=1 is compared to r2=1, compare occurs before the shift. After three transfers the loop is not taken anymore.

clock cycles total is 3*96 cycles=288 cycles
clock period = 1/100MHz=10 ns or 10*10^(-9) seconds
amount of transfer time = 288 cycles*(1/100MHz)=2.88us or 2.88*10^(-6) seconds
total amount of transferred data = 3*48=144 Bytes=1152 bits
bandwidth = (1152 bits)/(2.88us)=400 Mb/sec

5. Give the value of **`r0`** (in hex) after each instruction executes.

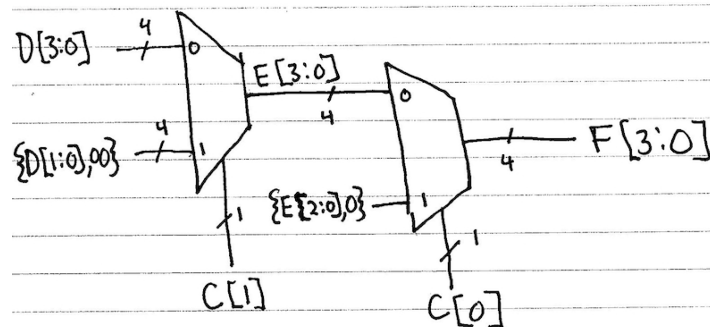
a) `mov r0, 0x1` ; `r0 =` ____**0x1**________

b) `mvn r0, 0x1` ; `r0 =` ____**0xFFFFFFFE**

c) `mov r0, 0x1, 11` ; `r0 =` ____**0x00200000**

d) `mvn r0, 0x1, 6` ; `r0 =` ____**0xFBFFFFFF**

6. Draw the logic diagram of a 4-bit barrel shifter that receives a 4-bit data value as input, **`D[3:0]`**, and a 2-bit control value, **`C[1:0]`**, as input and produces a 4-bit value output, **`F[3:0]`**. The output value is a left-shifted version of the input value and is shifted by the number of bits specified in **`C[1:0]`**. So, **`F[3:0]`**, can be left-shifted by 3, 2, 1, or 0 bits. The only logic parts you can use are 2:1 multiplexers with 4-bit data inputs and outputs. You can also use concatenation for the inputs. For example, if **`D[3:0]`**=$d_3d_2d_1d_0$, the concatenation indicated by **`{D[1:0],00}`** would be the bit string d_1d_0**`00`**. HINT: the posted class notes give an example of a 32-bit barrel shifter, so check them if you need help and re-familiarize yourself with the functionality of a multiplexer if you have forgotten how they operate.



C	E[3:0]	F[3:0]
0 0	$d_3d_2d_1d_0$	$d_3d_2d_1d_0$
0 1	$d_3d_2d_1d_0$	$d_2d_1d_0$ 0
1 0	d_1d_0 0 0	d_1d_0 0 0
1 1	d_1d_0 0 0	d_0 0 0 0