CS1022 Tutorial #0
SOLUTION Recap on CS1021

This tutorial has two aims: to see how much ARM assembly you remember and to refresh the content of CS1021. It is not marked and does not count towards grading in anyway. Rather than gathering and marking the tutorials, we will try using DirectPoll to collect answers. Each question has a link to its answer page.

1) Which of the following is the largest? **Link:**TEMP

   a) 70 Kib

   b) 70 kB

   c) 70 kb

   d) 70 KiB ✓

   e) Apollo 11's guidance computer's memory http://history.nasa.gov/computers/Ch2-5.html 32 kiloword, 16 bit word orgonised in to 1024 word banks so approx. 64kB (less usable as parity bits functions for extra instructions etc...)

   f) **extra info** Ki is the prefix for kibi. Kibi is 1024 (or 2^{10}). Lower case k is the SI kilo prefix it is 1000 (or 2^{10}). Upper case by is a byte, lower case b is a bit. There are 8 bits in a byte. Therefore, as Ki > k and B > b, KiB is the largest.

2) Which of the following code snippets will store the the sequence of integers $[17, 256]$ in memory. R0 is the address of some space, R1 is 17 and R2 is 0. **Link:**TEMP

a) ADD 4 as word

```
START
    CMP R2, R1
    BLT stop
    STR R1, [R0]
    ADD R0, R0, #1
    ADD R1, R1, #1
    B   START
```

b) ✓

```
START
    CMP R2, R1
    BLT stop
    STR R1, [R0]
    ADD R0, R0, #4
    ADD R1, R1, #1
    B   START
```

c) cant load 256 in byte

```
START
    CMP R2, R1
    BLT stop
    STRB    R1, [R0]
    ADD R0, R0, #1
    ADD R1, R1, #1
    B   START
```

d) ✓

```
START
    LDR R3, =1
    CMP R1, R2
    BGT stop
    STRH    R1, [R0]
    ADD R0, R0, R3, LSL #1
    ADD R1, R1, #1
    B   START
```

e) **extra info** in a) memory is byte addressable, so adding 1 moves on by one byte. STR stores a word which is 4 bytes, overwriting every three bytes, so counts in 4s. f) **extra info** in c doesn't work because 256 does not fit in a byte $2^8 - 1 = 255$

3) Which of the following code snippets will leave R0 with the highest value. R0 starts at −100 (0xFFFFFF9F) for each snippet.
Highest unsigned value **Link:**TEMP
Highest signed value **Link:**TEMP

a) 0xffffff06

```
ASR R0, R0, #4
EOR R0, R0, #0xFF
```

b) 0x0fffff06 high signed

```
LSR R0, R0, #4
EOR R0, R0, #0xFF
```

c) 0x0000fc00

```
LSL R0, R0, #3
AND R0, R0, #0xFF00
```

d) 0xffffffe0 high unsigned

```
LSL R0, R0, #3
ORR R0, R0, #0xFF00
```

e) 0xfbc10fc0

```
MUL R0, R0, R0 ;pretend that
MUL R0, R0, R0 ;this works
```

f)

```
LDR R0, =0
```

g) **extra info** ARM uses 2's compliment for signed numbers. This means the most significant bit is set for negative numbers. When negative signed unbers are treated as unsigned they are large.

4) Which of the following makes the condition code flags as Z = 0, V = 0, C = 0, N = 0 using SUBS and ADDS as appropriate? **Link:**TEMP

a) 0x80000000 + 0x80000000 z=1 v=1 c=1 n=0

b) 0x80000000 + 0x7FFFFFFF z=0 v=0 c=0 n=1 0xFFFFFFFF

c) 0xFFFFCFC7 + 0x00003039 z=1 v=0 c=1 n=0

d) 0x6E0074F2 + 0x211D6000 C=0, V=1, N=1, Z=0 0x8F1DD4F2

e) 0x00003039 − 0xFFFFCFC7 z=0 v=0 c=0 n=0 0x00006072

5) Write a program to convert some ASCII text stored in memory to uppercase. It should ignore whitespace and punctuation and leave uppercase letters alone.

See notes