Revision Question

- What is the value of R0, if R1 is 0 and if R1 is 1?

```
          CMP R1, #0
          BEQ SKIP
          ADD R0, R1, #5
    SKIP:
```

- *if R1 = 0, then R0 is zero, if R1 = 1, R) = 6.*
- Which condition flag does BLE (Branch if Less or Equal) check?
  Z OR (N XOR V) -> Zero for equal, negative without overflow for less.
- What is the value of R0, after executing the following code:
  r0 = 15 (5 + 4 + 3 + 2 + 1)

```
          MOV R0, #0
          MOV R1, #5
    LOOP: ADD R0, R0, R1
          SUBS R1, R1, #1
          BNE LOOP
```

Developing an application in ARM assembly

Overview

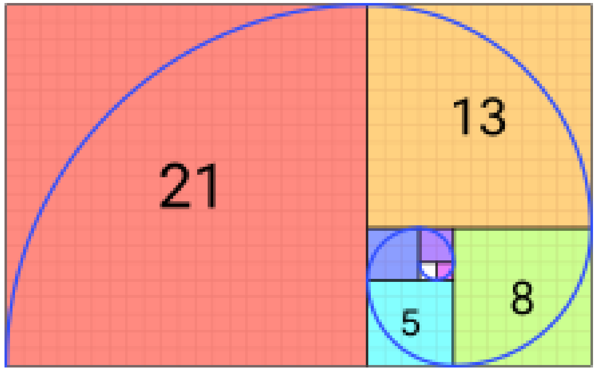
- We now have all the basic piece to implement programs in Assembly
  - Arithmetic & logical operators
  - Memory
  - Condition & Unconditional Loops

- Let’s put our experience in good use:
  - Fibonacci sequence
  - Letter capitalize

Fibonnacci sequence

Example program

- A program to create **Fibonacci** numbers
- Fibonacci numbers
  - A sequence of numbers of the following shape
    0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, …
  - Formula
    $F_n = F_{n-1} + F_{n-2}$ with $F_0 = 0$, $F_1 = 1$
- Requirements
  - To compute the first 12 numbers of the sequence
  - To store the resulting numbers in an array
  - At this point no I/O operation (show result on screen)



C Equivalent

- One option how to solve this in c (there are many options)
- No error checking here

```
int main(){
   int fibs[12] = {0};

   fibs[0] = 0;
   fibs[1] = 1;
   int n = 2;

   do {
      fibs[n] = fibs[n - 1] + fibs[n - 2];
      n = n + 1;
   } while (n < 12);
   return 0;
}
```

C Equivalent (alternative)

- One option how to solve this in c (there are many options)
- No error checking here

```
#define SIZE 12

int main(){
    int fibs[SIZE] = {0};

    fibs[0] = 0;
    fibs[1] = 1;

    for (int n = 2; n < SIZE; n = n + 1)
    {
        fibs[n] = fibs[n - 1] + fibs[n - 2];
    }

    return 0;
}
```

Assembler Steps

```
.data
fibs: .zero 48
```

Store 12 times a word (4byte) of content 0 and make the address of the first known as fibs

What comes next is for the data segment

```
int main(){
    int fibs[12] = {0};


    fibs[0] = 0;
    fibs[1] = 1;


    for (int n = 2; n < 12; n = n + 1)
    {
        fibs[n] = fibs[n - 1] + fibs[n - 2];
    }


    return 0;
}
```

Branch Example
for

```
int x = 0;
for (i=0;i < 10;i++) {
        x++;
}
```

```
        mov r0, #0
        mov r1, #0
WHILE:
        cmp r0, #10
        bge DONE
        add r0, r0, #1
        add r1, r1, #1
        b WHILE
DONE:
```

<u>register mapping</u>

```
i: r0
y: r1
```

Assembler Steps

```
.data
fibs: .word  0 : 12
.text
ldr r0, =fibs
mov r1, #0
str r1, [r0]
mov r1, #1
str r1, [r0, #4]
mov r1, 2
```

```
int main(){
   int fibs[12] = {0};

   fibs[0] = 0;
   fibs[1] = 1;

   for (int n = 2; n < 12; n = n + 1)
      fibs[n] = fibs[n - 1] + fibs[n - 2];

   return 0;
}
```

What comes next
is for the text segment

Mapping:
$t1 used for loop counter n
$t2 to store current fib number
$t3 for F[n-1], $t4 for F[n-2]

```
.data
fibs: .word  0 : 12
.text
ldr r0, =fibs
mov r1, #0
str r1, [r0]
mov r1, #1
str r1, [r0, #4]
mov r1, 2
add r0, 8
fib_loop:
ldr r2, [r0, -4]
ldr r3, [r0, -8]
add r2, r3
str r2, [r0]
add r1, 1
add r0, 4
cmp r1, #12
bne fib_loop
```

```c
int main(){
   int fibs[12] = {0};

   fibs[0] = 0;
   fibs[1] = 1;

   for (int n = 2; n < 12; n = n + 1)
      fibs[n] = fibs[n - 1] + fibs[n - 2];

   return 0;
}
```

Mapping:
r0 array pointer
r1 used for loop counter n
r2 to store current fib number
r2 for F[n-1], r3 for F[n-2]

```
.data
fibs: .zero 48          @ "array" F[ ] of 12 words


computeFibonacci:
    ldr r0, =fibs
    mov r1, #0
    str r1, [r0]
    mov r1, #1
    str r1, [r0, #4]
    mov r1, 2
    add r0, 8
fib_loop:
    ldr r2, [r0, -4]
    ldr r3, [r0, -8]
    add r2, r3
    str r2, [r0]
    add r1, 1
    add r0, 4
    cmp r1, #12
    bne fib_loop
```

String Capitalize

C program

```
const char *str = "Hello World!";

int main() {
  for (int i;i < 12;i++) {
    if ((str[i] < 'a') || (str[i] > 'z’))
      continue;
    str[i] = str[i] - 'a' + 'A’;
  }
}
```

Branch Example
if-then-else

```
if (g==h) f=g-h;
else f=g+h;
```

```
cmp r2, r3 @ r2 == r3
bne else
sub r0, r2, r3
b if_end
else:
add r0, r2, r3
if_end:
```

register mapping

f:	r0
g:	r2
h:	r3

```
.data
str: .asciz "Hello World!”
.text
capitalize:
    ldr r0, =str
    mov r1, #0
cap_loop:
    ldrb r2, [r0, r1]

    cmp r2, 'a’
    blt cap_skip
    cmp r2, 'z’
    bgt cap_skip

    sub r2, 'a’
    add r2, 'A’
    strb r2, [r0, r1]
cap_skip:
    add r1, 1
    cmp r1, #12
    bne cap_loop
```

```
const char *str = "Hello World!";

int main() {
 for (int i;i < 12;i++) {
  if ((str[i] < 'a') || (str[i] > 'z’))
   continue;
  str[i] = str[i] - 'a' + 'A’;
 }
}
```

```
.data
str: .asciz "Hello World!”
.text
capitalize:
    ldr r0, =str
    mov r1, #0
cap_loop:
    ldrb r2, [r0, r1]

    cmp r2, 'a’
    blt cap_skip
    cmp r2, 'z’
    bgt cap_skip

    sub r2, 'a’
    add r2, 'A’
    strb r2, [r0, r1]
cap_skip:
    add r1, 1
    cmp r1, #12
    bne cap_loop
```

Revision question

Sample C code.

```c
#include <stdio.h>

char string1[] = "HelloWorld",
  string2[] = "HelloWorld",
  string3[] = "HelloBorld";

int main() {
   int r9 = 0;
   for (int i = 0; i < 10; i++) {
      if (string1[i] != string2[i]) {
      r9 = 1;
      break;

  }
  return r9;
}
```

Fill in the following ARM code.

```
.syntax unified
.data

@ Let's define 3 strings for your program latter
string1: .asciz "HelloWorld"
string2: .asciz "HelloWorld"
string3: .asciz "HelloBorld"

.text

.global _start
_start:

@ example code
ldr r0, =string1
ldr r1, =string2

@ TODO: add code to compare r0 and r1. Store a zero
@ in r8, if strings are equal, or 1 otherwise. Your
@ solution can safely assume that strings have a
fixed
@ length of 10 characters. Swap r1 with label
string3 to
@ test if your code works if numbers are unequal.

b _start
```

Summary

Programming examples

Next

- Functions