# Why?

You’ll use the instructions you’ve already learned about to construct loops and array accesses by a variable index.

# Model 1: Loop and array access

|  |  |  |  |
| --- | --- | --- | --- |
|  | C program | ARM program A | ARM program B |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | int fibs[20];  int f1 = 1;  int f2 = 1;  int i = 0;  while (i < 20) {  int f3 = f1 + f2;  int f1 = f2;  int f2 = f3;  fibs[i] = f3;  i++;  } | mov R0, #0x00002B00  mov R1, #1  mov R2, #1  mov R12, #0  frog:  cmp R12, #20  bge parrot  add R3, R1, R2  mov R1, R2  mov R2, R3  lsl R7, R12, #2  str R3, [R0, R7]  add R12, R12, #1  b frog  parrot: | mov R0, #0x00002B00  mov R1, #1  mov R2, #1  add R12, R0, #80  turkey:  cmp R0, R12  bge lion  add R3, R1, R2  mov R1, R2  mov R2, R3  str R3, [R0], #4  b turkey  lion: |

1. Match the lines of C code with the lines of its translation in ARM program A.

1 – 1

2 – 2

3 – 3

4 – 4

I < 20 – 6

6 – 8

7 – 9

8 – 10

9 – 11 & 12

10 - 13

1. Match the lines of C code with the lines of its translation in ARM program B.

1 – 1

2 – 2

3 – 3

4 – 4

5 I < 20 – 6

6 – 8

7 – 9

8 – 10

9 – 11

1. Explain how the while loop condition is translated in ARM program A.

R12 it basically just an index number, it is seeing if it is equal to the number 20.

1. Describe the specific purpose of register R12 in making program B work.

R12 is the last index of the array. It gets to this by multiplying the number of numbers in the array (20) by the number each int takes up (4) to get 80.

1. Explain how the while loop condition is translated in ARM program B.

It compares R0 to R12, R0 being the current index of the array, and R12 being the last index of the array.

1. Generalize: explain how you translate a loop from C into ARM.

You just keep going back to a branch, which checks the condition each time at the start of it.

Branch:

Cmp

Bne Out

…body

B Branch

Out:

# Read This!

Any loop in assembly language requires a ***back edge***, that is, a branch that goes to an earlier instruction.

1. Why would a back edge be required to make a loop?

A back edge is the whole essence of a loop, repeating code over and over. So, back edges are required because you can go back to earlier code and repeat it.

1. Explain the specific purpose of the lsl instruction in making ARM program A work.

Lsl gives a pointer to a specific index of the array.

1. Why doesn’t ARM program B require the lsl instruction?

Because R0 (the memory address) is being updated by 4 (the size of what is being stored) (therefore moving to the next integer of the array) every time *after* it is stored.

1. ARM program A has an instruction to increment “i” in order to make progress in the loop. How does ARM program B make progress in the loop?

It adds 4 to the memory address every time.

1. Generalize: Describe two different ways to access an array by a variable index in ARM.

Index (A): You can use an index (1-20) and use lsl to get a pointer to a spot in an array by multiplying the index number by 4 (the size of what is being stored) and the address. – compute the address &array[i] = &array[0] + S\*i

Address (B): Increment by using the address and incrementing by however much the storage size of what you are storing is. – keeping the &array[i] in a register and incrementing it each iteration.

# Read This!

The str instructions in program A and program B use different ***indexing modes***. Program A uses ***offset*** mode and program B uses ***post-index*** mode.

1. What might be the advantages and disadvantages of offset mode and post-index mode?