# Why?

You are now familiar with reading and writing ARM assembly language programs. These programs are human-readable, but in order for the computer to understand them, we need a binary representation of instructions. In this packet, you will consider how ARM instructions may be encoded with bits.

In the context of the computing stack, your practice translating high level programs to assembly language programs took the place of the compiler, and now translating assembly language programs to machine code takes the place of the assembler.

**Manager:** Total expected time to complete Model 1-5: 150 minutes. All questions are important, but if you are running behind, then skip the Extension questions.

**Quality Controller:** Make sure your group has ARM reference sheets in front of you for this activity.

# Model 1: A binary encoding for a small instruction set

Consider the following made-up instruction set.

* has only these registers, named: R0, R1, R2, R3
* has only these instructions
  + add Rd, Rn Rd <- Rd + Rn
  + ldr Rd, [Rn, imm] Rd <- Mem[Rn + imm]
  + str Rd, [Rn, imm] Mem[Rn + imm] <- Rd
* imm (short for *immediate*) means an integer constant in the range [ -27, 27-1 ]

1. How many bits do we need to name each instruction name (add, ldr, str) uniquely? Give each its own binary number.

|  |  |
| --- | --- |
| **instruction name** | **binary number** |
| add | 01 |
| ldr | 10 |
| str | 11 |

1. How many bits do you need to name each register uniquely? Give each its own binary number.

|  |  |
| --- | --- |
| **register name** | **binary number** |
| R0 | 00 |
| R1 | 01 |
| R2 | 10 |
| R3 | 11 |

1. How many registers are involved in a given instruction?

two

1. How many bits are needed for the immediate?

four

1. Summarize your previous answers in this table.

|  |  |
| --- | --- |
| **Component** | **bits to specify it** |
| instruction name | 2 |
| register Rd | 2 |
| register Rn | 2 |
| immediate | 8 |

1. How many bits are required to specify the whole instruction?

14 – fixed

6 - variable

1. Use your previous answers to translate each of these instructions into a single binary number. Make sure all three are translated the same way.
   1. add R3, R1
      1. 01 11 01
   2. ldr R1, [R2, #4]
      1. 10 01 10 0000-0100
   3. str R0, [R3, #-8]
      1. 11 00 11 1111-1000

# Read this!

The binary version of assembly language code is called ***machine code***.

1. In complete sentences, describe how you translate an instruction in assembly language to machine code.

Just put in order of whatever you wrote in ARM in binary.