In a nutshell : Every computer has a *binary machine language* , in which instructions are written as series of 0's and 1's, and a *symbolic machine language* , also known as *assembly language* , in which instructions are expressed using human-friendly mnemonics. Both languages do exactly the same thing, and are completely equivalent. But, writing programs in assembly is far easier and safer then writing in binary. In order to enjoy this luxury, someone has to translate our symbolic programs into binary code that can execute as-is on the target computer. This translation service is done by an agent called *assembler* . The assembler can be either a person who carries out the translation manually, or a computer program that automates the process. In this module and final project in the course we learn how to build an assembler. In particular, we'll develop the capability of translating symbolic Hack programs into binary code that can be executed as-is on the Hack platform. Each one of you can choose to accomplish this feat in two different ways: you can either implement an assembler using a high-level language, or you can simulate the assembler's operation using paper and pencil. In both cases we give detailed guidelines about how to carry out your work.

Key concepts: Binary and symbolic machine languages, parsing, symbol tables, code generation, cross assembler, assembler implementation.

WATCH the module 6 lectures:

- Unit 6.1: Assembly Languages and Assemblers
- Unit 6.2: The Hack Assembly Language
- Unit 6.3: The Assembly Process - Handling Instructions
- Unit 6.4: The Assembly Process - Handling Symbols
- Unit 6.5: Developing a Hack Assembler
- Unit 6.6: Project 6 Overview: Programming Option
- Unit 6.6B: Project 6 Overview: Without Programming
- Unit 6.7: Perspectives

DO:

You have two options for project 6. A programming option, and an option for non-programmers.

Programming option:

1. Complete Project 6: The Assembler .

2. Use your assembler to translate the .asm programs found in the nand2tetris/projects/06/ folder to .hack programs.
3. Submit the following files: Add.hack, Max.hack, Rect.hack, Pong.hack (Make sure you submit the files created by **your** assembler). In your zip file, include a file named "prog.txt" (its contents don't matter).

   Non-programming option:

1. Using the techniques taught in the lectures, and as explained in unit 6.6B, you should know how to translate .asm files to .hack files by hand.
2. Translate the following two files: MaxL.asm, and Rect.asm, which you can find in the nand2tetris/projects/06/ folder on your computer.
3. Submit MaxL.hack and Rect.hack (Be sure to submit the files you created by hand). In your zip file, include a file named "hand.txt" (its contents don't matter).

   If you are taking the course as an auditor, you can check your work yourself, using the tests described here . If you are taking the certificate option, submit your project zip file here .

GET HELP in the Module 6 Discussion Forum