Assembler Tutorial

This program is part of the software suite that accompanies

*The Elements of Computing Systems*
by Noam Nisan and Shimon Schocken

MIT Press

[www.nand2tetris.org](http://www.nand2tetris.org)

This software was developed by students at the Efi Arazi School of Computer Science at IDC

Chief Software Architect: Yaron Ukrainitz
The Elements of Computing Systems evolves around the construction of a complete computer system, done in the framework of a 1- or 2-semester course.

In the first part of the book/course, we build the hardware platform of a simple yet powerful computer, called Hack. In the second part, we build the computer’s software hierarchy, consisting of an assembler, a virtual machine, a simple Java-like language called Jack, a compiler for it, and a mini operating system, written in Jack.

The book/course is completely self-contained, requiring only programming as a pre-requisite.

The book’s web site includes some 200 test programs, test scripts, and all the software tools necessary for doing all the projects.
The book’s software suite

(All the supplied tools are dual-platform: Xxx.bat starts Xxx in Windows, and Xxx.sh starts it in Unix)

Simulators (HardwareSimulator, CPUEmulator, VMEmulator):
- Used to build hardware platforms and execute programs;
- Supplied by us.

Translators (Assembler, JackCompiler):
- Used to translate from high-level to low-level;
- Developed by the students, using the book’s specs; Executable solutions supplied by us.

Bin: simulators and translators software;
- builtIn: executable versions of all the logic gates and chips mentioned in the book;
- os: executable version of the Jack OS;
- TextComparer: a text comparison utility.

The machine code generated by the assembler can be tested either in the hardware simulator or in the CPU emulator.
I. Assembly program example

II. Command-level Assembler

III. Interactive Assembler

Relevant reading: Chapter 4: *Machine and Assembly Language*
Part I:
Assembly Programming at a Glance
Example

**Sum.asm**

```assembly
// Computes sum=1+...+100.
@i    // i=1
M=1
@sum  // sum=0
M=0
(LOOP)
@i    // if (i-100)=0 goto END
D=M
@100
D=D-A
@END
D;JGT
@i     // sum+=i
D=M
@sum
M=D+M
@i     // i++
D=M+1
@LOOP // goto LOOP
0;JMP
(END)   // infinite loop
@END
0;JMP
```

**Sum.hack**

```
0000000000010000
1110111111001000
0000000000010001
1110101010001000
0000000000010000
1111100000010000
0000000000010001
1111000010001000
0000000000010000
1111110000010000
0000000000010001
1111000010001000
0000000000010000
1110000000000000
0000000000000001
1110000000000000
0000000000000001
1111100000000000
0000000000000001
1111000001000000
0000000000010000
1110101010001000
0000000000010000
1110101010001000
0000000000010000
0000000000000000
```

Assembler Tutorial, www.nand2tetris.org
Example

**Sum.asm**

```assembly
// Computes sum=1+...+100.
@i    // i=1
M=1
@sum  // sum=0
M=0
(LOOP)
    @i    // if (i-100)=0 goto END
    D=M
    @100
    D=D-A
    @END
    D;JGT
    @i     // sum+=i
    D=M
    @sum
    M=D+M
    @i     // i++
    M=M+1
    @LOOP  // goto LOOP
    0;JMP
(END)    // infinite loop
@END
0;JMP
```

The assembly program:
- Stored in a text file named **Prog.asm**
- Written and edited in a text editor

The assembly process:
- Translates **Prog.asm** into **Prog.hack**
- Eliminates comments and white space
- Allocates variables (e.g. **i** and **sum**) to memory
- Translates each assembly command into a single 16-bit instruction written in the Hack machine language
- Treats label declarations like (**LOOP**) and (**END**) as pseudo commands that generate no code.
Part II:

Learn how to invoke the supplied assembler from the OS shell level.

(the assembler that you have to write in project 6 should have the same GUI and behavior)
The command-level assembler

Display the assembly source code (contents of the .asm text file)

We illustrate how to use the assembler in the Windows command level (DOS); The Unix way is similar.
Inspecting the source file

```
G:\>cd progs\Sum
G:\progs\Sum>type Sum.asm
// Computes sum=1+...+100.
@i   // i=1
M=1
@sum // sum=0
M=0

<LOOP>
@i   // if <i-100)=0 goto END
D=M
@100
D=D-A
@END
D;JGT
@i   // sum+=i
D=M
@sum
M=D+M
@i   // i++
M=M+1
@LOOP // goto LOOP
0;JMP

<END>  // infinite loop
@END
0;JMP

G:\progs\Sum>
```
Invoking the Assembler

Invoke the assembler program

Name of the file to be translated (argument of the assembler program).
Invoking the Assembler

Two ways to test the generated machine code:
1. Invoke the hardware simulator, load the `Computer.hdl` chip, then load the code (.hack file) into the internal ROM chip;
2. Load and run the code in the CPU emulator (much quicker).
Part III:
Learn how to use the interactive Assembler
Loading an assembly program

Navigate to a directory and select an .asm file.
Loading an assembly program

- Read-only view of the assembly source code
- To edit it, use an external text editor.
Translating a program

- Translate line-by-line
- Translate the entire program
- Pause the translation
- Start from the beginning
- Immediate translation (no animation)
Inspecting the translation

1. Click an assembly command
2. The corresponding translated code is highlighted
Saving the translated code

- The “save” operation is enabled only if the translation was error-free;
- Otherwise, the translation stops with an error message.
Using Compare Files

1. Load a compare file

2. Select a compare (.hack) file
Using Compare Files

1. Compare file is shown

2. Translate the program (any translation mode can be used)
Using Compare Files

The translation of the highlighted line does not match the corresponding line in the compare file.
On weekends, my father would take me for walks in the woods and he’d tell me about interesting things that were going on. “See that bird?” he says. “It’s a Spencer Warbler.” (I knew he didn’t know the real name.) “Well, in Italian, it’s Chutto Lapittida. In Portuguese, it’s a Bom da Peida. In Chinese, it’s a Chung-long-tah, and in Japanese, it’s Katano Tekeda. You can know the name of that bird in all the languages of the world, but when you’re finished, you’ll know absolutely nothing whatever about the bird. You’ll only know something about people in different places, and what they call the bird. So let’s look at the bird and see what it is doing - that’s what counts.” This is how I learned very early the difference between knowing the name of something and knowing something.