# Assignment 2 Description

## Assignment 2 - Translating Virtual Machine Code

## Weighting and Due Dates

- Marks for this assignment contribute 10% of the overall course mark.
- Marks for functionality will be awarded automatically by the web submission system.
- **Due dates: Milestone** - **11:55pm Tuesday of week 9**, **Final** - **11:55pm Friday of week 9**.
- **Late penalties:** For each part, the maximum mark awarded will be reduced by 25% per day / part day late. If your mark is greater than the maximum, it will be reduced to the maximum.
- **Core Body of Knowledge (CBOK) Areas:** abstraction, design, hardware and software, data and information, and programming.

## Project Description

In this assignment you will complete a variation of projects 7 and 8 in the nand2tetris course. You will complete a program, **translator**, which will read a Hack Virtual Machine file and output a Hack Assembly Language translation of each Virtual Machine command. The output will be in a form suitable for testing by the precompiled executable program **bin/simulator**.

### SVN Repository

**Note**: this assignment assumes that you have already created directories for every assignment, workshop, project and exam in your svn repository, as described on the **Startup Files for Workshops and Assignments (https://myuni.adelaide.edu.au/courses/54311/pages/startup-files-for-workshops-and-assignments)** page.

1. If required, checkout a working copy of the assignment2 directory from your svn repository.
2. Change directory to the working copy of the assignment2 directory.
3. Copy the newest zip file attached below into the updates sub-directory and add it to svn.
4. Run the following command to place the assignment's startup files in the correct locations:

```
% make install
```

5. Add the tokensier.cpp file and an empty tests directory to your svn repository:

```
% svn add translator.cpp
% svn add --depth=empty tests
% svn commit -m "Assignment 2 Startup Files"
```

## Assignment 2 Files and Directories

In addition to the generic **Makefile** and **updates** sub-directory, the assignment2 directory should now contain the following files and directories:

- **translator.cpp** - C++ source file.
- **translator** - executable script that will run your compiled **translator** program.
- **bin** - this directory contains precompiled programs and scripts.
- **includes** - this directory contains **.h** files for the library.
- **lib** - this directory contains precompiled library components.
- **originals** - this directory contains the original version of the **translator.cpp**.
- **tests** - this directory contains test data, you can add your own tests here.

**Note**: you need to edit the **translator.cpp** file to complete this assignment. All the other files are automatically regenerated every time you run **make**, they must not be changed or added to **svn**. You can add extra test inputs to the **tests** directory but those are the only additional files that you may add to **svn**.

**Note**: if a newer version of the startup files is made available, it must be placed in the **updates** sub-directory and added to **svn**. The next time **make** is run, all of the files will be updated except for **transator.cpp**.

# Submission and Marking Scheme

Submissions for this assignment must be made to the **web submission system (https://cs.adelaide.edu.au/services/websubmission)** *assignment* named: ***Assignment 2 - Submit Here***. The assessment is based on "**Assessment of Programming Assignments (https://myuni.adelaide.edu.au/courses/54311/pages/assessment-of-programming-assignments)**".

**Note:** the ***Submit Here*** assignment will show a breakdown of your marks by category but it will always show your total mark as **0**.

**Your programs must be written in C++**. Your programs will be compiled using the **Makefile** included in the zip file attached below. The **translator** program will be compiled using the file **translator.cpp** file. No other **.cpp** or **.h** files should be present in your **svn** directory. Your programs will then be tested using a small set of Hack Virtual Machine files. You may assume that the first command in each test will be a function command so that you will always know the correct class and function name to use when constructing labels.

**Note**: the simulator tests each virtual machine command implementation separately so the overall failure of a test could still result in a significant number of marks for those individual virtual machine command implementations that behaved correctly.

# Assignment 2 - Milestone Submissions: due 11:55pm Tuesday of week 9

The marks awarded by the **web submission system (https://cs.adelaide.edu.au/services/websubmission)** for the milestone submission contribute up to 20% of your marks for assignment 2. Your milestone submission mark, after the application of late penalties, will be posted to the myuni gradebook when the assignment marking is complete.

## Automatic Marking

The Milestone Tests are the same as the Final Tests but, marks will only be recorded for those tests that require a complete implementation of the following virtual machine commands:

- function
- push constant
- push static
- pop pointer
- add
- sub
- neg
- and
- or
- not

You can view the Milestone Tests marks in the **Milestone** *assignment* but submissions must be made using the ***Assignment 2 - Submit Here*** *assignment.*

# Assignment 2 - Final Submissions: due 11:55pm Friday of week 9

The marks awarded for the final submission contribute up to 80% of your marks for assignment 2.

Your final submission mark will be the geometric mean of three components, the marks for the Final Tests, a mark for your logbook and a mark for your code. It will be limited to 20% more than the marks for the Final Tests. See "**Assessment - Mark Calculations (https://myuni.adelaide.edu.au/courses/54311/pages/assessment-mark-calculations)** " for examples of how the marks are combined. Your final submission mark, after the application of late penalties, will be posted to the myuni gradebook when the assignment marking is complete.

**NOTE - A logbook mark of 0 results in a Final Submission mark of 0.**

You can view the Final Tests marks in the **Final** *assignment* but submissions must be made using the ***Assignment 2 - Submit Here*** *assignment.*

## Logbook Marking

**Important**: the logbook must have entries for all work in this assignment, including your milestone submissions. See "**Assessment - Logbook Review (https://myuni.adelaide.edu.au/courses/54311/pages/assessment-logbook-review)** " for details of how your logbook will be assessed.

## Code Review Marking

For each of your programming assignments you are expected to submit well written **code**. See "**Assessment - Code Review (https://myuni.adelaide.edu.au/courses/54311/pages/assessment-code-review)** " for details of how your code will be assessed.

# Assignment 2 - Participation Marks

Any submissions to assignment 2 that are made before the due date for Milestone Submissions may be awarded up to 10 participation marks. The participation marks will be the marks awarded for the Final Tests divided by 10. You can view the participation marks awarded in the ***Pre Milestone*** *assignment* but submissions must be made using the ***Assignment 2 - Submit Here*** *assignment*. The participation marks will be allocated to week 8.

# Virtual Machine Translator

# Background

This assignment requires you to complete the implementation of the Hack Virtual Machine **translator** program. You should have started working out how to implement VM commands in Hack Assembler in workshop 07. It would be a good idea to revisit workshop 07 before continuing with this assignment.

The **translator.cpp** file contains a recursive descent parser that will recognise Hack VM commands and call a C++ function to output a Hack Assembly language implementation of each kind of VM command it parses. You are required to edit the **translator.cpp** file and complete each of the translation functions. Do not modify the other parts of the **translator.cpp** file.

The translation functions must output Hack Assembly language by calling the function, **output_assembler()**, which is passed a single line of Hack Assembly language or a comment. Each translation function starts and ends with start_* and end_* functions so that program output will be in a form suitable for testing by the precompiled executable program **bin/simulator**.

Remember to keep your logbook up to date when working on this assignment and make svn commits when you add new logbook entries.

# Startup Files

The newest of the following zip file(s) must be placed in the updates sub-directory and added to svn. When make is run, the newest zip file in the updates directory is used to update the startup files. Any files you are required to edit will not be updated but, a copy of the latest version of those files will be placed in the sub-directory originals.

- **assignment2-20200923-135641.zip (https://myuni.adelaide.edu.au/courses/54311/files/7490998/download?wrap=1)**
- assignment2-20200923-120412.zip