Hack assembly fragments *

Jens Lechtenbörger

IT Systems, Summer Term 2025

1 Hack assembly exercises

If you really want to understand this, please enter the instructions in the CPU Emulator provided by Nand2Tetris. Execute them and observe their effects on registers and RAM locations. (Maybe also effects on the screen, to be explained later: Enter different numbers into RAM[16384] and observe pixel changes towards the top left of the screen...)

1.1 First exercises

- Set A to 17

```
@17
```

- Set D to A-1

```
D=A-1
```

- Set both A and D to A + 1

```
AD=A+1
```

- Set D to 19

```
@19
D=A
```

- Set RAM[5034] to D - 1

```
@5034
M=D-1
```

- Set RAM[53] to 171

```
@171
D=A
@53
M=D
```

- Load RAM[7], add 1, and store the result in D

```
@7
D=M+1
```

- Increment the number stored in the RAM location whose address is stored in RAM[42]

```
@42
A=M
M=M+1
```

*This PDF document is an inferior version of an OER in HTML format; free/libre Org mode source repository.

1.2 Variables

If you write the code examples below to files (whose default endings should be `.asm` for assembler code), variable names are resolved into RAM locations by a software called assembler.

While we do not look into the inner working of the assembler, you can load such `asm` files into the CPU Emulator, which also replaces variable names (and other symbols) with appropriate numbers. For example, it will allocate variables to RAM locations starting with RAM[16]. Note that you cannot enter variable names directly in the CPU Emulator.

- sum = 0

```
@sum
M=0
```

- j = j + 1

```
@j
M=M+1
```

- q = sum + 12 - j

```
@sum
D=M
@12
D=D+A
@j
D=D-M
@q
M=D
```

License Information