**Hand Compiling Hack Assembly Language**

University of the People

CS1104: Computer Systems

Instructor, Thomas Skoff

Wednesday, July 27, 2022

Hand Compiling Hack Assembly Language

The following assignment involves two Hack assembly programs. The task is to translate each into binary machine code. Then load the machine language program into the CPU simulator and execute it. A description of my process to convert the assembler symbols into machine language and what I learned in the process is included with the assembler programs and corresponding machine language programs.

The Hack microinstructions below were used to decode the instructions:

Table

Description automatically generated

**Program 1**

**Hack Assembly Language**

//Program 1 to convert to machine language  
// Computes R0 = 2 + 3  
  
@2  
D=A  
@3  
D=D+A  
@0  
M=D

**Process**

@2  
D=A

A-instruction: @2 is equivalent to 0000 0000 0000 00102

C-instruction: D=A, D is the destination, making the 3 destination bits = 010. The computation is equal to A, making the 6 computation bits = 110000 and the a=0. No jump is indicated, making the 3 jump bits = 000. 🡪 1110 1100 0001 00002

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *1* | *x1* | *x2* | *a* |  | *c1* | *c2* | *c3* | *c4* |  | *c5* | *c6* | *d1* | *d2* |  | *d3* | *j1* | *j2* | *j3* |
| *1* | *1* | *1* | *0* |  | *1* | *1* | *0* | *0* |  | *0* | *0* | *0* | *1* |  | *0* | *0* | *0* | *0* |

@3  
D=D+A

A-instruction: @3 is equivalent to 0000 0000 0000 00112

C-instruction: D=D+A, D is the destination, making the 3 destination bits = 010. The computation is equal to D+A, making the 6 computation bits = 000010 and the a=0. No jump is indicated, making the 3 jump bits = 000. 🡪 1110 0000 1001 00002

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *1* | *x1* | *x2* | *a* |  | *c1* | *c2* | *c3* | *c4* |  | *c5* | *c6* | *d1* | *d2* |  | *d3* | *j1* | *j2* | *j3* |
| *1* | *1* | *1* | *0* |  | *0* | *0* | *0* | *0* |  | *1* | *0* | *0* | *1* |  | *0* | *0* | *0* | *0* |

@0  
M=D

A-instruction: @0 is equivalent to 0000 0000 0000 00002

C-instruction: M=D, M is the destination, making the 3 destination bits = 001. The computation is equal to D, making the 6 computation bits = 001100 and the a=0. No jump is indicated, making the 3 jump bits = 000. 🡪 1110 0011 0000 10002

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *1* | *x1* | *x2* | *a* |  | *c1* | *c2* | *c3* | *c4* |  | *c5* | *c6* | *d1* | *d2* |  | *d3* | *j1* | *j2* | *j3* |
| *1* | *1* | *1* | *0* |  | *0* | *0* | *1* | *1* |  | *0* | *0* | *0* | *0* |  | *1* | *0* | *0* | *0* |

**Binary Machine Code**

0000000000000010

1110110000010000

0000000000000011

1110000010010000

0000000000000000

1110001100001000

**CPU Emulator**

**Program 2**

**Hack Assembly Language**

// Program 2 to convert to machine language  
// Symbol-less version of the Max.asm program.

@0

D=M  
@1

D=D-M  
@10

D;JGT  
@1

D=M  
@12

0;JMP  
@0

D=M  
@2

M=D  
@14

0;JMP

**Process**

@0

D=M

A-instruction: @0 is equivalent to 0000 0000 0000 00002

C-instruction: D=M, D is the destination, making the 3 destination bits = 010. The computation is equal to M, making the 6 computation bits = 110000 and the a=1. No jump is indicated, making the 3 jump bits = 000. 🡪 1111 1100 0001 00002

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *1* | *x1* | *x2* | *a* |  | *c1* | *c2* | *c3* | *c4* |  | *c5* | *c6* | *d1* | *d2* |  | *d3* | *j1* | *j2* | *j3* |
| *1* | *1* | *1* | *1* |  | *1* | *1* | *0* | *0* |  | *0* | *0* | *0* | *1* |  | *0* | *0* | *0* | *0* |

@1 D=D-M

A-instruction: @1is equivalent to 0000 0000 0000 00012

C-instruction: D=D-M, D is the destination, making the 3 destination bits = 010. The computation is equal to D-M, making the 6 computation bits = 010011 and the a=1. No jump is indicated, making the 3 jump bits = 000. 🡪 1111 0100 1101 00002

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *1* | *x1* | *x2* | *a* |  | *c1* | *c2* | *c3* | *c4* |  | *c5* | *c6* | *d1* | *d2* |  | *d3* | *j1* | *j2* | *j3* |
| *1* | *1* | *1* | *1* |  | *0* | *1* | *0* | *0* |  | *1* | *1* | *0* | *1* |  | *0* | *0* | *0* | *0* |

@10

D;JGT

A-instruction: @10 is equivalent to 0000 0000 0000 10102

C-instruction: D;JGT, D is the destination, making the 3 destination bits = 010. The computation is equal to D, making the 6 computation bits = 001100 and the a=0. JGT indicates if the output is greater than 0 jump, making the 3 jump bits = 001. 🡪 1110 0011 0001 00012

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *1* | *x1* | *x2* | *a* |  | *c1* | *c2* | *c3* | *c4* |  | *c5* | *c6* | *d1* | *d2* |  | *d3* | *j1* | *j2* | *j3* |
| *1* | *1* | *1* | *0* |  | *0* | *0* | *1* | *1* |  | *0* | *0* | *0* | *1* |  | *0* | *0* | *0* | *1* |

When I ran the code in the Assembler, I got this binary code:

1110 0011 000 00001

This means I did not interpret the destination correctly. So, null must be the destination.

@1

D=M

A-instruction: @1 is equivalent to 0000 0000 0000 00012

C-instruction: D=M, D is the destination, making the 3 destination bits = 010. The computation is equal to M, making the 6 computation bits = 110000 and the a=1. No jump is indicated, making the 3 jump bits = 000. 🡪 1111 1100 000100002

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *1* | *x1* | *x2* | *a* |  | *c1* | *c2* | *c3* | *c4* |  | *c5* | *c6* | *d1* | *d2* |  | *d3* | *j1* | *j2* | *j3* |
| *1* | *1* | *1* | *1* |  | *1* | *1* | *0* | *0* |  | *0* | *0* | *0* | *1* |  | *0* | *0* | *0* | *0* |

@12

0;JMP

A-instruction: @12 is equivalent to 0000 0000 0000 11002

C-instruction: 0;JMP, indicates a go to loop. Null is the destination, making the 3 destination bits = 000. The computation is equal to 0, making the 6 computation bits = 101010 and the a=0. JMP is indicated, making the 3 jump bits = 111. 🡪 1110 1010 1000 01112

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *1* | *x1* | *x2* | *a* |  | *c1* | *c2* | *c3* | *c4* |  | *c5* | *c6* | *d1* | *d2* |  | *d3* | *j1* | *j2* | *j3* |
| *1* | *1* | *1* | *0* |  | *1* | *0* | *1* | *0* |  | *1* | *0* | *0* | *0* |  | *0* | *1* | *1* | *1* |

@0

D=M (repeated)

A-instruction: @0 is equivalent to 0000 0000 0000 00002

C-instruction: D=M, D is the destination, making the 3 destination bits = 010. The computation is equal to M, making the 6 computation bits = 110000 and the a=1. No jump is indicated, making the 3 jump bits = 000. 🡪 1111 1100 0001 00002

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *1* | *x1* | *x2* | *a* |  | *c1* | *c2* | *c3* | *c4* |  | *c5* | *c6* | *d1* | *d2* |  | *d3* | *j1* | *j2* | *j3* |
| *1* | *1* | *1* | *1* |  | *1* | *1* | *0* | *0* |  | *0* | *0* | *0* | *1* |  | *0* | *0* | *0* | *0* |

@2

M=D

A-instruction: @2 is equivalent to 0000 0000 0000 00102

C-instruction: M=D, M is the destination, making the 3 destination bits = 001. The computation is equal to D, making the 6 computation bits = 001100 and the a=0. No jump is indicated, making the 3 jump bits = 000. 🡪 1110 0011 0000 10002

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *1* | *x1* | *x2* | *a* |  | *c1* | *c2* | *c3* | *c4* |  | *c5* | *c6* | *d1* | *d2* |  | *d3* | *j1* | *j2* | *j3* |
| *1* | *1* | *1* | *0* |  | *0* | *0* | *1* | *1* |  | *0* | *0* | *0* | *0* |  | *1* | *0* | *0* | *0* |

@14

0;JMP

A-instruction: @14 is equivalent to 0000 0000 0000 11102

C-instruction: 0;JMP, indicates a go to loop. Null is the destination, making the 3 destination bits = 000. The computation is equal to 0, making the 6 computation bits = 101010 and the a=0. JMP is indicated, making the 3 jump bits = 111. 🡪 1110 1010 1000 01112

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *1* | *x1* | *x2* | *a* |  | *c1* | *c2* | *c3* | *c4* |  | *c5* | *c6* | *d1* | *d2* |  | *d3* | *j1* | *j2* | *j3* |
| *1* | *1* | *1* | *0* |  | *1* | *0* | *1* | *0* |  | *1* | *0* | *0* | *0* |  | *0* | *1* | *1* | *1* |

**Binary Machine Code**

0000000000000000

1111110000010000

0000000000000001

1111010011010000

0000000000001010

1110001100000001

0000000000000001

1111110000010000

0000000000001100

1110101010000111

0000000000000000

1111110000010000

0000000000000010

1110001100001000

0000000000001110

1110101010000111

**Conclusion**

What I learned from this assignment, is that I do know how to properly decode Hack instructions. I did have a problem decoding D;JGT. Why does the D not decode into binary? I will research further to determine the reasoning.

References

Nisan, N., & Schocken, S. (2005). *The elements of computing systems: Machine Learning.* MIT Press.  [**http://f.javier.io/rep/books/The%20Elements%20of%20Computing%20Systems.pdf**](http://f.javier.io/rep/books/The%20Elements%20of%20Computing%20Systems.pdf)