David Duffrin Project 7

October 18, 2016

1 Project 7: A Virtual Machine

In [50]:
```
# The Parser Module
import Project7IO as IO

line = ""
nexttype= ""
nextarg1=""
nextarg2=""

def hasMoreCommands():
    global line
    if line == "EOF":
        return False
    else:
        return True

def advance():
    global nexttype, nextarg1, nextarg2,line
    line = IO.nextLine()
    if line == "EOF":
        return

    pieces = line.split()
    numpieces = len(pieces)
    nexttype = pieces[0]
    if numpieces == 1:
        nextarg1 = ''
        nextarg2 = ''
    elif numpieces == 2:
        nextarg1 = pieces[1]
        nextarg2 = ''
    else:
        nextarg1 = pieces[1]
        nextarg2 = pieces[2]
```

```
        def commandType():
            global nexttype
            if nexttype == "push":
                return 'C_PUSH'
            elif nexttype == "pop":
                return 'C_POP'
            else:
                return 'C_ARITHMETIC'

        def arg1():
            global nexttype, nextarg1
            if commandType() == 'C_ARITHMETIC':
                return nexttype
            else:
                return nextarg1

        def arg2():
            global nextarg2
            return nextarg2
```

In [168]:
```
# Testing the Parser

 def partadvance(line):
     global nexttype, nextarg1, nextarg2
     pieces = line.split()
     numpieces = len(pieces)
     nexttype = pieces[0]
     if numpieces == 1:
         nextarg1 = ''
         nextarg2 = ''
     elif numpieces == 2:
         nextarg1 = pieces[1]
         nextarg2 = ''
     else:
         nextarg1 = pieces[1]
         nextarg2 = pieces[2]

 def check(a,b,c):
     assert commandType() == a
     assert arg1() == b
     assert arg2() == c

 line = 'push constant 0'
 partadvance(line)
 check('C_PUSH', 'constant','0')

 line = 'pop local 1'
```

```
partadvance(line)
check('C_POP','local','1')

line = 'add'
partadvance(line)
check('C_ARITHMETIC','add','')
```

In [166]:
```
jumpcounter = 0
filename = ""
cnt = 0

def writeArithmetic(command):
    global cnt
    print('@SP')
    if command == 'neg' or command == 'not':
        print('A=M-1')
        if command == "neg":
            print('M=-M')
        elif command == "not":
            print('M=!M')
    else:
        print('AM=M-1')
        print('D=M')
        if command in ['add','sub','and','or']:
            print('A=A-1')
            if command == "add":
                print('M=M+D')
            elif command == "sub":
                print('M=M-D')
            elif command == "and":
                print('M=M&D')
            elif command == "or":
                print('M=M|D')
        else:
            print('@SP')
            print('AM=M-1')
            print('A=M')
            print('D=A-D')
            print('@JMP' + str(cnt))
            cnt += 1
            print("D;J" + command.upper())
            print('@SP')
            print('A=M')
            print('M=0')
            print("@JMP" + str(cnt))
            cnt += 1
            print('0;JMP')
            print('(JMP' + str(cnt - 2) + ')')
```

```
            print('@SP')
            print('A=M')
            print('M=-1')
            print('(JMP' + str(cnt - 1) + ')')
            print('@SP')
            print('M=M+1')

    def writePushPop(type, segment, index):
        if segment == "constant":
            print("@" + index)
            print('D=A')
        elif segment == "local":
            print("@LCL")
            print("D=M")
            print("@" + index)
        elif segment == "argument":
            print("@ARG")
            print("D=M")
            print("@" + index)
        elif segment == "this":
            if type == 'push':
                print('@THIS')
                print("D=M")
                print("@" + index)
            else:
                print("@THIS")
                print("D=M")
                print("@" + index)
                print("D=D+A")
                print("@R13")
                print("M=D")
                print("@SP")
                print("AM=M-1")
                print("D=M")
                print("@R13")
                print("A=M")
                print("M=D")
        elif segment == "that":
            if type == 'push':
                print('@THAT')
                print("D=M")
                print("@" + index)
            else:
                print("@THAT")
                print("D=M")
                print("@" + index)
                print("D=D+A")
                print("@R13")
```

```
                print("M=D")
                print("@SP")
                print("AM=M-1")
                print("D=M")
                print("@R13")
                print("A=M")
                print("M=D")
        elif segment == "pointer":
            if index == '0':
                print("@THIS")
            else:
                print("@THAT")
            if type == "pop":
                print("D=A")
            else:
                print("D=M")
        elif segment == "temp":
            print("@R5")
            print("D=M")
            print("@" + str(int(index) + 5))
        elif segment == "static":
            print("@STATIC." + index)
            print("D=M")
            print("@" + index)

        if type == "push":
            if segment != "constant" and segment != "pointer":
                print("A=D+A")
                print("D=M")
            print("@SP")
            print("A=M")
            print("M=D")
            print("@SP")
            print("M=M+1")

        if type == "pop" and segment != 'this' and segment != 'that':
            if segment != "pointer":
                print("D=D+A")
            print("@R13")
            print("M=D")
            print("@SP")
            print("AM=M-1")
            print("D=M")
            print("@R13")
            print("A=M")
            print("M=D")
```

In [167]:
```
#The Main Module
```

```
import os

def processFile(testtype, fname):
    global line
    IO.setFile(os.path.join('..',testtype,fname,fname+'.vm'))
    IO.setSaveFile(os.path.join('..',testtype,fname,fname+'.asm'))
    line = ""
    advance()
    while hasMoreCommands():
        if commandType() == "C_ARITHMETIC":
            writeArithmetic(arg1())
        elif commandType() == "C_PUSH":
            writePushPop("push", arg1(), arg2())
        elif commandType() == "C_POP":
            writePushPop("pop", arg1(), arg2())
        advance()

processFile('MemoryAccess', 'BasicTest')

#Uncomment these lines once you pass the Basic Test.
processFile('MemoryAccess', 'PointerTest')
processFile('MemoryAccess', 'StaticTest')

#Uncomment these lines once you have handled memory regions as well.
processFile('StackArithmetic', 'SimpleAdd')
processFile('StackArithmetic', 'StackTest')
```

```
//BasicTest.asm
@10
D=A
@SP
A=M
M=D
@SP
M=M+1
@LCL
D=M
@0
D=D+A
@R13
M=D
@SP
AM=M-1
D=M
@R13
A=M
M=D
@21
D=A
@SP
A=M
M=D
@SP
M=M+1
@22
D=A
@SP
A=M
M=D
@SP
M=M+1
@ARG
D=M
@2
D=D+A
@R13
M=D
@SP
AM=M-1
D=M
@R13
A=M
```

```
//PointerTest.asm
@3030
D=A
@SP
A=M
M=D
@SP
M=M+1
@THIS
D=A
@R13
M=D
@SP
AM=M-1
D=M
@R13
A=M
M=D
@3040
D=A
@SP
A=M
M=D
@SP
M=M+1
@THAT
D=A
@R13
M=D
@SP
AM=M-1
D=M
@R13
A=M
M=D
@32
D=A
@SP
A=M
M=D
@SP
M=M+1
@THIS
D=M
@2
```

```
//StaticTest.asm
@111
D=A
@SP
A=M
M=D
@SP
M=M+1
@333
D=A
@SP
A=M
M=D
@SP
M=M+1
@888
D=A
@SP
A=M
M=D
@SP
M=M+1
@STATIC.8
D=M
@8
D=D+A
@R13
M=D
@SP
AM=M-1
D=M
@R13
A=M
M=D
@STATIC.3
D=M
@3
D=D+A
@R13
M=D
@SP
AM=M-1
D=M
@R13
A=M
```

```
//SimpleAdd.asm
@7
D=A
@SP
A=M
M=D
@SP
M=M+1
@8
D=A
@SP
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
A=A-1
M=M+D
```

```
//StackTest.asm
@17
D=A
@SP
A=M
M=D
@SP
M=M+1
@17
D=A
@SP
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
@SP
AM=M-1
A=M
D=A-D
@JMP0
D;JEQ
@SP
A=M
M=0
@JMP1
0;JMP
(JMP0)
@SP
A=M
M=-1
(JMP1)
@SP
M=M+1
@17
D=A
@SP
A=M
M=D
@SP
M=M+1
@16
D=A
@SP
```

```
M=D
@ARG
D=M
@1
D=D+A
@R13
M=D
@SP
AM=M-1
D=M
@R13
A=M
M=D
@36
D=A
@SP
A=M
M=D
@SP
M=M+1
@THIS
D=M
@6
D=D+A
@R13
M=D
@SP
AM=M-1
D=M
@R13
A=M
M=D
@42
D=A
@SP
A=M
M=D
@SP
M=M+1
@45
D=A
@SP
A=M
M=D
@SP
M=M+1
@THAT
```

```
D=D+A
@R13
M=D
@SP
AM=M-1
D=M
@R13
A=M
M=D
@46
D=A
@SP
A=M
M=D
@SP
M=M+1
@THAT
D=M
@6
D=D+A
@R13
M=D
@SP
AM=M-1
D=M
@R13
A=M
M=D
@THIS
D=M
@SP
A=M
M=D
@SP
M=M+1
@THAT
D=M
@SP
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
A=A-1
M=M+D
```

```
M=D
@STATIC.1
D=M
@1
D=D+A
@R13
M=D
@SP
AM=M-1
D=M
@R13
A=M
M=D
@STATIC.3
D=M
@3
A=D+A
D=M
@SP
A=M
M=D
@SP
M=M+1
@STATIC.1
D=M
@1
A=D+A
D=M
@SP
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
A=A-1
M=M-D
@STATIC.8
D=M
@8
A=D+A
D=M
@SP
A=M
M=D
@SP
```

```
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
@SP
AM=M-1
A=M
D=A-D
@JMP2
D;JEQ
@SP
A=M
M=0
@JMP3
0;JMP
(JMP2)
@SP
A=M
M=-1
(JMP3)
@SP
M=M+1
@16
D=A
@SP
A=M
M=D
@SP
M=M+1
@17
D=A
@SP
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
@SP
AM=M-1
A=M
D=A-D
@JMP4
```

```
D=M
@5
D=D+A
@R13
M=D
@SP
AM=M-1
D=M
@R13
A=M
M=D
@THAT
D=M
@2
D=D+A
@R13
M=D
@SP
AM=M-1
D=M
@R13
A=M
M=D
@510
D=A
@SP
A=M
M=D
@SP
M=M+1
@R5
D=M
@11
D=D+A
@R13
M=D
@SP
AM=M-1
D=M
@R13
A=M
M=D
@LCL
D=M
@0
A=D+A
D=M
@THIS
D=M
@2
A=D+A
D=M
@SP
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
A=A-1
M=M-D
@THAT
D=M
@6
A=D+A
D=M
@SP
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
A=A-1
M=M+D
@SP
M=M+1
@SP
AM=M-1
D=M
A=A-1
M=M+D
D;JEQ
@SP
A=M
M=0
@JMP5
0;JMP
(JMP4)
@SP
A=M
M=-1
(JMP5)
@SP
M=M+1
@892
D=A
@SP
A=M
M=D
@SP
M=M+1
@891
D=A
@SP
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
@SP
AM=M-1
A=M
D=A-D
@JMP6
D;JLT
@SP
A=M
M=0
@JMP7
0;JMP
(JMP6)
@SP
A=M
M=-1
(JMP7)
@SP
```

```
@SP
A=M
M=D
@SP
M=M+1
@THAT
D=M
@5
A=D+A
D=M
@SP
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
A=A-1
M=M+D
@ARG
D=M
@1
A=D+A
D=M
@SP
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
A=A-1
M=M-D
@THIS
D=M
@6
A=D+A
D=M
@SP
A=M
M=D
@SP
M=M+1
@THIS
D=M
M=M+1
@891
D=A
@SP
A=M
M=D
@SP
M=M+1
@892
D=A
@SP
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
@SP
AM=M-1
A=M
D=A-D
@JMP8
D;JLT
@SP
A=M
M=0
@JMP9
0;JMP
(JMP8)
@SP
A=M
M=-1
(JMP9)
@SP
M=M+1
@891
D=A
@SP
A=M
M=D
@SP
M=M+1
@891
D=A
@SP
A=M
```

```
@6
A=D+A
D=M
@SP
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
A=A-1
M=M+D
@SP
AM=M-1
D=M
A=A-1
M=M-D
@R5
D=M
@11
A=D+A
D=M
@SP
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
A=A-1
M=M+D
```

```
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
@SP
AM=M-1
A=M
D=A-D
@JMP10
D;JLT
@SP
A=M
M=0
@JMP11
0;JMP
(JMP10)
@SP
A=M
M=-1
(JMP11)
@SP
M=M+1
@32767
D=A
@SP
A=M
M=D
@SP
M=M+1
@32766
D=A
@SP
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
@SP
AM=M-1
A=M
D=A-D
@JMP12
D;JGT
```

```
@SP
A=M
M=0
@JMP13
0;JMP
(JMP12)
@SP
A=M
M=-1
(JMP13)
@SP
M=M+1
@32766
D=A
@SP
A=M
M=D
@SP
M=M+1
@32767
D=A
@SP
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
@SP
AM=M-1
A=M
D=A-D
@JMP14
D;JGT
@SP
A=M
M=0
@JMP15
0;JMP
(JMP14)
@SP
A=M
M=-1
(JMP15)
@SP
M=M+1
```

```
@32766
D=A
@SP
A=M
M=D
@SP
M=M+1
@32766
D=A
@SP
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
@SP
AM=M-1
A=M
D=A-D
@JMP16
D;JGT
@SP
A=M
M=0
@JMP17
0;JMP
(JMP16)
@SP
A=M
M=-1
(JMP17)
@SP
M=M+1
@57
D=A
@SP
A=M
M=D
@SP
M=M+1
@31
D=A
@SP
A=M
M=D
```

```
@SP
M=M+1
@53
D=A
@SP
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
A=A-1
M=M+D
@112
D=A
@SP
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
A=A-1
M=M-D
@SP
A=M-1
M=-M
@SP
AM=M-1
D=M
A=A-1
M=M&D
@82
D=A
@SP
A=M
M=D
@SP
M=M+1
@SP
AM=M-1
D=M
A=A-1
M=M|D
@SP
```

				A=M-1 M=!M