הנחות:

1. קלט תקין - כמות חוקית של שורות ומשתנים

משתנים גלובליים:

Global Symbol table = {(name, value),…} //dictionary, of Case sensitive variables.

//keys are strings

//values are integers

ארכיטקטורה ראשית:

**Initialize()**

1. Create the global table()
2. Insert 23 pre-defined values.
3. //

**Main(XXX.asm)**

1. Initialize()
2. Open XXX.asm
3. lablesProccessor()
4. varProccessor()
5. parser()
6. return XXX.hack
7. //יותר עבודה לכתוב בעברית הערה.

**labelsProccessor(XXX.asm) // only “(BlaBlaBla)” is interesting**

1. iterate the file
2. recognize labels
3. enter labels to the symbol table
4. no return value
5. //לזהות מה מוקף בסוגריים

**varProccessor(XXX.asm) // only @ are interesting.**

1. iterate the file
2. recognize variables
3. enter to symbol table
4. create tempFile.asm
5. write tempFile.asm with no symbols // output is only “pure” A code or C code
6. return tempFile.asm

**parser(tempFile.asm)**

1. iterate tempFile.asm lines.
2. Create XXX.hack
3. For each line:
   1. Skip empty lines or comments lines
   2. Recognize per each line if it is C code or A code and perform one out of two options:
      1. BinaricLine = codeAcode(line)
      2. BinaricLine = codeCcode(line)
   3. Write the line into XXX.hack
4. Return the XXX.hack

**codeAcode(A code line)**

//convert and return A code into binaric code (is there a package?)

codeCcode(C code line)

//recognize and separate the destination part, comp part and jump part

//(remember there is an option of default value for each of the options)

// convert C code into binary (by calling the 3 assisting function)

// Concat the binary parts together and return

**codeJumpInstruction()**

//According to the table

**codeDestInstruction()**

//According to the table

**codeCompInstruction()**

//According to the table