the CSC 258 Computer

- up to 64 Megabytes of memory, 4 bytes per word, word addressable
- 32-bit arithmetic (2's complement integers, IEEE single-precision floating-point)
- accumulator, 32 bits (AC)
- condition code, 1 bit (E)
- input port, 8-bit parallel
- output port, 8-bit parallel

Instructions

LDA m:	(LoaD Accumulator) Load the contents of memory location m into AC.
STA m:	(STore Accumulator) Store the contents of AC at memory location m.
ADD m:	(ADD) Integer Add the contents of memory location m to the contents of AC and put the sum in AC; if overflow, E is 1, otherwise E is 0.
SUB m:	(SUBtract) Integer Subtract the contents of memory location m from the contents of AC and put the difference in AC; if overflow, E is 1, otherwise E is 0.
MUL m:	(MULtilpy) Integer Multiply the contents of AC by the contents of memory location m and put the product in AC; if overflow, E is 1, otherwise E is 0.
DIV m:	(DIVide) Integer Divide the contents of the accumulator by the contents of memory location m and put the quotient in AC; if divisor is 0, E is 1, otherwise E is 0.
MOD m:	(MODulo) Integer Divide the contents of the accumulator by the contents of memory location m and put the remainder in AC; if divisor is 0, E is 1, otherwise E is 0.
FLA m:	(FLoating Add) Floating Add the contents of memory location m to the contents of AC and put the sum in AC; if overflow, E is 1, otherwise E is 0.
FLS m:	(FLoating Subtract) Floating Subtract the contents of memory location m from the contents of AC and put the difference in AC; if overflow, E is 1, otherwise E is 0.
FLM m:	(FLoating Multilpy) Floating Multiply the contents of AC by the contents of memory location m and put the product in AC; if overflow, E is 1, otherwise E is 0.
FLD m:	(FLoating Divide) Floating Divide the contents of the accumulator by the contents of memory location m and put the quotient in AC; if divisor is 0 or overflow, E is 1, otherwise E is 0.
CIF m:	(Convert Integer to Float) Load the contents of memory location m into AC and convert from integer to float.
CFI m:	(Convert Float to Integer) Load the contents of memory location m into AC and convert from float to the nearest integer.
AND m:	(AND) AND each bit of AC with the corresponding bit of memory location m and put the result in AC; if the result is all 0s, E is 0, otherwise E is 1.
IOR m:	(Inclusive OR) OR each bit of AC with the corresponding bit of memory location m and put the result in AC; if the result is all 0s, E is 0, otherwise E is 1.
XOR m:	(eXclusive OR) XOR each bit of AC with the corresponding bit of memory location m and put the result in AC; if the result is all 0s, E is 0, otherwise E is 1.
BUN m:	(Branch UNconditional) Branch to memory location m.

BZE m:	(Branch on Zero E) If E is 0, branch to memory location m.
BSA m:	(Branch and Save Address) Store the address of the next instruction (i.e., following this instruction) at memory location m, and branch to memory location m + 1.
BIN m:	(Branch INdirect) Branch to the memory location whose address is contained in memory location m.
INP m:	(INPut) If the input port has a byte of information ready, put it in the low-order (rightmost) eight bits of AC and 0s in the high-order 24 bits; otherwise branch to memory location m.
OUT m:	(OUTput) If the output port is ready to receive a byte of information, send the low-order eight bits of AC; otherwise branch to memory location m.

Data

I 123	Data word, integer format
F -1.2	Data word, floating-point format
C 'abc'	Data word, character format, in reverse order padded on left with 0s to 4 characters
B 1010	Data word, binary format, padded on left with 0s to 32 bits
H 5BF2	Data word, hexadecimal format, padded on left with 0s to 8 digits
A there	Data word, address (identifier or number)
W 123	The specified number of data words, with no initialization

An Assembly Language program consists of a sequence of lines; each line says what's in one word of memory (except for the W data format). A line begins with zero or more labels; each label is an identifier followed by a colon. After that, the line is either in instruction format or in data format. Instruction format is an op-code followed by an address. The op-code is three letters, and the address is either a number or an identifier. Data format is one of I (for 2's complement integer), F (for IEEE floating point number), C (for ASCII character), B (for binary), H (for hexadecimal), A (for address), or W (for word) followed by the indicated data. The W format reserves the indicated number of words of memory. The remainder of any line can be used for comments. W 0 can be used just for a label, or just for a comment. Here is a sequence of example lines (this is not a sensible program).

```
 main: LDA x
       ADD c3
       STA y
       BUN there
    x: I   0
    y: F   -1.2e3
   c3: I   3
there: BUN opsys
```

Execution starts at label `main` . Execution of your program ends with a branch to the operating system at address `opsys`. All identifiers used as addresses, except `opsys`, should be defined as labels. An identifier cannot be defined more than once as a label.