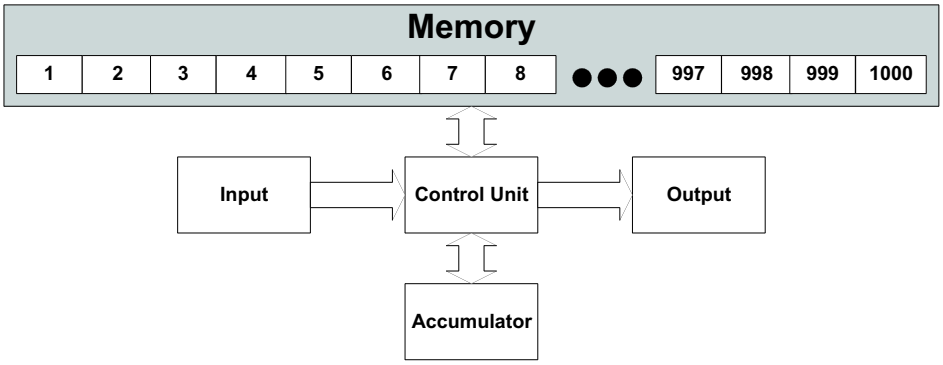
Problem 3	Grandpa's Computer	6 Points

Program Name: grandpa.cpp **Input File: grandpa.dat**

The earliest computers had very simple architectures in which there was a simple accumulator on which all computations were performed. The computations (and memory) were very limited in nature. Your program will simulate the earliest versions of the assembly language operations for one of these computers. The following picture illustrates the architecture of the computer you will emulate.



The components of the computer are as follows.

- **Input** – The computer has an input mechanism which reads (from something called paper tape) a stream of sequential values. These values are your computer's program. (Hint: there will be no loops.)
- **Accumulator** –A single "memory" location called a register that is the destination of many operations including all mathematical operations.
- **Memory** – This computer has 1000 memory locations (pretty advanced, huh?) in which it can store 32-bit integers (hey, we can dream).
- **Output** – A sequential stream output to a paper tape that your program will simulate.
- **Control Unit** – The heart of the computer's processing capability. This unit reads your program from Input and executes the instructions it reads.

The computer can execute the following instructions:

Operation	Instruction Format	Function
Load a value from memory into the accumulator.	LOAD <mem>	ACC ← MEMORY[mem]
Store the value in the accumulator into memory.	STOR <mem>	MEMORY[mem] ← ACC
Data initialization of the accumulator.	DATA <val>	ACC ← <val>
Output the accumulator's contents	OUT	ACC → OUTPUT
Add memory contents to accumulator.	ADD <mem>	ACC ← ACC + MEMORY[mem]
Subtract memory contents from accumulator.	SUB <mem>	ACC ← ACC - MEMORY[mem]
Multiply memory contents by accumulator.	MUL <mem>	ACC ← ACC * MEMORY[mem]
Divide memory contents into accumulator.	DIV <mem>	ACC ← ACC / MEMORY[mem]
Modulus memory contents into accumulator.	MOD <mem>	ACC ← ACC % MEMORY[mem]

Some notes about the operation of the computer.

- All operations are based on 32-bit integers only and any division operation results in an integer result.
- Any divide-by-zero will result in a value of zero being stored in the accumulator. This applies to both the divide and modulus operations.
- The initial contents of the accumulator and all memory locations will be the value zero.
- The input stream is "preprocessed" to ensure that there are no invalid instructions or memory locations for those operations that involve a memory location.

Input
For this problem, your program should begin reading the contents of the input stream which will be stored one instruction per line. Each line of input will contain exactly one of the valid instructions from the table above. The instruction will start in column one. If the operation is the OUT operation, the operation will be immediately followed by the end-of-line character. All other operation will be followed by exactly on space character and then a single integer value and then the end-of-line character. If the operation calls for a memory location, the integer will be between 1 and 1000; otherwise, the integer will be a single signed 32-bit integer. There will be no invalid input in the stream.

Output
Output from your program will be the result of each OUT operation. Each OUT operation will print the integer value of the accumulator on a line by itself to the screen.

Example: Input File
```
DATA 100
STOR 3
DATA 3
STOR 4
LOAD 3
ADD 4
STOR 5
LOAD 3
SUB 4
STOR 6
LOAD 3
MUL 4
STOR 7
LOAD 3
DIV 4
STOR 8
LOAD 3
MOD 4
STOR 9
LOAD 3
OUT
LOAD 4
OUT
LOAD 5
OUT
LOAD 6
OUT
LOAD 7
OUT
LOAD 8
OUT
LOAD 9
OUT
```
Output to screen
```
100
3
103
97
300
33
1
```