poopmachine documentation

Version 1.0

Revision 1

Preface

poopmachine is an emulated virtual processor running on a Reduced Instruction Set Computer (RISC) architecture. **poop**machine was made as a side project and is not designed for useful computations. At its current state, **poop**machine is not Turing complete, however, there are plans to make it so in future versions.

System architecture and organisation

poopmachine is a register machine with a fixed number of registers. **poop**machine does not possess Random Access Memory or any other external memory. The program that **poop**machine runs is stored in actual memory (i.e. not in virtual **poop**machine memory) and cannot be modified after initial input. The only form of memory available for use in computation are the four registers in **poop**machine. Arithmetic operations are centred around the main register, known as the accumulator, and all data is stored as signed integers. Programs for **poop**machine move on to the next line once the previous set of instructions has been executed.

Registers

Register	Name	Description
ACC	Accumulator	Main register for operations. Arithmetic operations are performed on this register.
DAT	Data	Additional register for data storage. Contents may be swapped with the accumulator.
PC	Program Counter	Special register which stores the current line of the program **poop**machine is on. Line numbers start from 0. Auto-incrementing. When the value of PC exceeds the highest line number of the program, PC is set to (PC VALUE) mod (highest line number). When the value of PC is negative, PC is set to 0.
CMS	Comparison Status	Special register which stores the result of comparison instructions. (Not implemented yet)

I/O pseudo-registers

Keyword	Description
IN	Gets value from user input. -1 is returned if use input is invalid. Blocking (will halt entire program until user input is received)and read only
OUT	Values written to it are sent to the output Write only

Instruction set

Instruction	Syntax	Description
NOP	NOP	Does nothing. (Temporary solution for jumps-use before a label if not program will break)
ADD	ADD <VAL/REG>	ACC is increased by the value Program is terminated and returns error if OUT is used
SUB	SUB <VAL/REG>	ACC is decreased by the value Program is terminated and returns error if OUT is used
NEG	NEG <REG>	The value in the register is negated Undefined behaviour if IN or OUT is used
SWP	SWP	The values in ACC and DAT are swapped
MOV	MOV <DST> <SRC>	Value of SRC is copied onto DST. Program is terminated and returns error if IN is used for DST Program is terminated and returns error if OUT is used for SRC
LABEL	<LABEL>:	Sets a label at that line to jump to, converted to NOP upon processing
JMP	JMP <LABEL>	Value of PC is set to the number of the line that LABEL is on. (Effectively

		jumping to LABEL, see above for known-bug)

Program input modes

Dynamic

Program is entered via the terminal line by line. Can also be copied and pasted. Once entered, previous lines cannot be edited.

File

broke do not use

Program is read from a text file containing the program.