Computer Science Students

Instructors Corner

Simple assembler virtual machine

Virtual machine is an interpreter for assembly language, so it acts as a physical computer. Assembly language is language for a specific computer architecture. In this case, the specific computer architecture is virtual but this should be transparent to the user.

The computer is a simple computer with ALU, RAM, just one register called ACCumulator, simple IO, and a built-in stack. Word size is 2 bytes, and the addressability is thus 64k. The computer uses standard data 2's complement data representation so thus data range is -32k to +32k. See AccVM.pdf.

Format

- Each line
  - independent and self-contained, and may be blank, an instruction, or a storage directive
  - all delimiters are WS
- Instructions
  - for an accumulator machine

(left argument and result are in an implicit accumulator ACC register, except for COPY), with the following format

XXX arguments

- XXX is the reserved name, required, in upper case
- arguments as needed separated by spaces
- additional optional label can start any instruction
  label:
  just one label per line at most
- instruction list (# arguments, meaning)

- ADD (1, ACC = ACC +arg)
- BR (1, jump to arg)
- BRNEG (1, jump to arg if ACC <0)
- BRZNEG (1, jump to arg if ACC <=0)
- BRPOS (1, jump to arg if ACC >0)
- BRZPOS (1, jump to arg if ACC >=0)
- BRZERO (1, jump to arg if ACC ==0)
- COPY (2, arg1 = arg2)
- DIV (1, ACC = ACC / arg)
- MULT (1, ACC = ACC * arg)
- READ (1, arg=input integer)
- WRITE (1, put arg to output as integer)
- STOP (0, stop program)
- STORE (1, arg = ACC)
- SUB (1, ACC = ACC - arg)
- NOOP (0, nothing)
- LOAD (1, ACC=arg)

ADD, DIV, MULT, WRITE, LOAD, SUB can take either variable or immediate value

as the arg: immediate value is positive integer or negative integer

- PUSH (0, tos++)
- POP (0, tos--)
- STACKW (1,stack[tos-arg]=ACC)
- STACKR (1,ACC=stack[tos-arg])

PUSH/POP are only means to reserve/delete automatic storage.
STACKW/STACKR n - these are stack write/read instructions.
n must be a non-negative number, and the access is to nth element down from TOS

NOTE: TOS points to the topmost element on the stack

- Storage directives

XXX val

- XXX is a name
- val is the initial value
- all storage and ACC size are signed 2 bytes

- Storage name and label are all names starting with latter and following with letters and digits up to eight total

Semantics

- execution begins with the first line and continues until STOP is reached

Assumptions

- any proper format within line, tokens separated by WS
- all storage directives are listed following the last STOP
- all names start with letters and contain more letters or digits

Location

`/accounts/classes/janikowc/cs4280/asmInterpreter/virtMach`

- readable, also download here as admiral executable virtMach

Invocation

> `virtMach` // read from stdin

> `virtMach file.asm` // read from file.asm

Example:

- `sumOf3.asm`
  - reads 3 arguments and returns the sum using a stack

```
READ X
PUSH
LOAD X
STACKW 0

READ X
PUSH
LOAD X
STACKW 0

READ X
STACKR 1
```

```
STACKR 1
ADD X
STORE X
STACKR 0
ADD X
STORE X
WRITE X
POP
POP
STOP
X 0
```

Example:

- sum3nostack.asm
  - same as above but without a stack

```
READ X
READ Y
READ Z
LOAD X
ADD Y
ADD Z
STORE X
WRITE X
STOP
X 0
Y 0
Z 0
```

Example:

- sumOfAny.asm
  - reads first argument and the reads as many arguments as the first argument and returns the sum

```
      READ X
      COPY Z X

LOOP1: LOAD X
      BRZERO OUT1
      BRNEG  OUT1
      READ Y
      LOAD Y
      PUSH
```

```
      STACKW 0
      LOAD X
      SUB 1
      STORE X
      BR LOOP1

OUT1:  NOOP

      LOAD 0
      STORE Y
LOOP2: STACKR 0
      ADD Y
      STORE Y
      POP
      LOAD Z
      SUB 1
      BRZERO OUT2
      BRNEG OUT2
      STORE Z
      BR LOOP2
OUT2:  NOOP
      WRITE Y

STOP
X 0
Y 0
Z 0
```

Edited by: Cezary Janikow 11 months ago

Tags: None Edit

Viewer Comments

Contributors (1)

Activity (5)