S0 Processor

S0 is an accumulator machine, i.e. it has only one register (one-address instruction format). Its aim is to be the tool for teaching assembly language. The idea is to simplify many details in machine instructions founded in modern processors such as: register assignment, complex addressing mode and subroutine call. All variables are global resided in the main memory (not in registers). The instruction format is very simple. It has only one format, one-argument instructions of the following form:

op:8 arg:24

where op is the instruction code, arg is the argument which is an unsigned integer 24-bit. The instruction set is as follows:

arithmetic and logic :add, sub, inc, dec, eq, ne, lt, le, gt, ge
control: jmp, jt, jf
data: ld, st

The following is the meaning of each instruction:

```
arith & logic        ac = ac op M[arg]
inc                  ac = M[arg] + 1
dec                  ac = M[arg] - 1
```

with logical instructions, the result is true/false where false is 0, true is non-zero.

```
jmp                  pc = arg
jt  (jmp if true)    if ac != 0 pc = arg
jf  (jmp if false)   if ac == 0 pc = arg
ld  (load)           ac = M[arg]
st  (store)          M[arg] = ac
```

Instruction encoding

```
1 add   2 sub    3 inc   4 dec   5 eq
6 ne    7 lt     8 le    9 gt    10 ge
11 jmp  12 jt    13 jf   14 ld   15 st
```

With one-address instructions, all constants must be in the memory. There is no indirect addressing hence access to data structures is not possible (no array etc.)

Assembly language

There are two sections in the assembly language source: data section and code section. The data section contains all the symbolic names declaration (for variable names) associated with their values. The code section contains instructions.

```
.data
name value
...
.end
```

```
.code
[:label] op arg
...
.end
```

A label is used for jump destination in the jump instructions (jmp, jt, jf). A label is prefixed with ":" except when it is used as an argument. The argument of an instruction is a name except for the jump instruction, the argument is a label. By default, the code section is started at the memory address 0, the data section starts at the address 1000.

Here is an example program to sum 1..10.

a pseudo code

```
i = 1
s = 0
while i <= 10
  s = s + i
  i = i + 1
```

S0 assembly language

```
.data
i 1
s 0
ten 10
.end

.code
:loop
  ld i
  le ten
  jf exit
  ld s
  add i
  st s
  inc i
  st i
  jmp loop
:exit
.end
```

Prabhas Chongstitvatana
18 December 2006