

Custom Language to Assembly Compiler

Program takes input either from file or from keyboard and outputs .asm files.

- Main file – Takes input and produces .asm file
- Scanner – Reads characters and produces tokens, checks for syntax errors
- Parser – Interprets tokens
- Semantics – Checks for semantic errors

Lexical Definitions

- Alphabet
 - all English letters (upper and lower), digits, plus the extra characters as seen below, plus WS
 - No other characters allowed and they should generate errors
- Identifiers
 - begin with a letter and
 - continue with any number of letters or digits
 - you may assume no identifier is longer than 8 characters (in testing)
- Keywords (reserved, suggested individual tokens)
 - begin end loop void var return in out program iffy then let data
- Operators and delimiters group (all single character except ==)
 - = < > == : + - * / % . () , { } ; []
- Integers
 - any sequence of decimal digits, no sign, no decimal point

Semantics


- Basic semantics as in C - program executes sequentially from the beginning to the end, one statement at a time
- Conditional statement is like the else-less if statement in C
- Loop statement is like the while loop in C
- Assignment evaluates the expression on the right and assigns to the ID on the left
- +* are standard arithmetical, / is integer division, unary * is negation
- All expressions are evaluated before being used
- Relational and arithmetical operators have the standard meaning except:
 - < and > are less than and greater than
 - < < is less equal
 - > > is greater equal
 - < > is NOT equal
 - == is equal
- IO reads/prints a 2-byte signed integer
- All data is 2-byte signed integer

BNF

```
<program> -> <vars> <block>
<block>   -> begin <vars> <stats> end
<vars>    -> empty | data Identifier = Integer . <vars>
<expr>    -> <N> - <expr> | <N>
<N>       -> <A> / <N> | <A> * <N> | <A>
<A>       -> <M> + <A> | <M>
<M>       -> * <M> | <R>
<R>       -> ( <expr> ) | Identifier | Integer
<stats>    -> <stat> <mStat>
<mStat>    -> empty | <stat> <mStat>
<stat>     -> <in> . | <out> . | <block> | <if> . | <loop> . | <assign> .
<in>       -> in Identifier
<out>      -> out <expr>
<if>       -> iffy [ <expr> <RO> <expr> ] then <stat>
<loop>     -> loop [ <expr> <RO> <expr> ] <stat>
<assign>   -> Identifier = <expr>
<RO>      -> < | < < (two tokens) | > | > > (two tokens) | == (one token ==) | < > (two tokens)
```

Example Input

```
data x = 1 .
data y = 2 .
begin
  in x .
  loop [ x > 0 ]
  begin
    y = x .
    loop [ y > 0 ]
    begin
      out y .
      y = y - 2 .
    end .
    x = x - 1 .
  end .
out y .
end
```



Example Output

```
LOAD 1
STORE x
LOAD 2
STORE y
READ x
L1: NOOP
LOAD 0
STORE T0
LOAD x
SUB T0
BRZNEG L0
LOAD x
STORE y
L3: NOOP
LOAD 0
STORE T1
LOAD y
SUB T1
BRZNEG L2
LOAD y
STORE T2
WRITE T2
LOAD 2
```

```
STORE T3
LOAD y
SUB T3
STORE y
BR L3
L2: NOOP
LOAD 1
STORE T4
LOAD x
SUB T4
STORE x
BR L1
L0: NOOP
LOAD y
STORE T5
WRITE T5
STOP
T0 0
T1 0
T2 0
T3 0
T4 0
T5 0
x 0
y 0
```

Home

Documentation

Run

Next Instruction

Reset Runtime

Clear Editor

Upload

Download

Examples

ASM Editor

```

35 BR L1
36 L0: NOOP
37 LOAD y
38 STORE T5
39 WRITE T5
40 STOP
41 T0 0
42 T1 0
43 T2 0
44 T3 0
45 T4 0
46 T5 0
47 x 0
48 y 0
49

```

Output:

Please enter value for x

5

OK

Cancel

Assembler Settings

Pop-out Documentation

Accumulator

0

Variables

Variable	Value
T0	0
T1	0
T2	1
T3	2
T4	1
T5	-1
x	0

Stack

Top of the Stack is index 0

Position	Value
2	3
1	4
0	2
-1	1
-2	1
-3	2
-4	1
-5	2
-6	1
-7	2
-8	1
-9	2
-10	1
-11	2
-12	1
-13	2
-14	1
-15	2
-16	1
-17	2
-18	1
-19	2
-20	1
-21	2
-22	1
-23	2
-24	1
-25	2
-26	1
-27	2
-28	1
-29	2
-30	1
-31	2
-32	1
-33	2
-34	1
-35	2
-36	1
-37	2
-38	1
-39	2
-40	1
-41	2
-42	1
-43	2
-44	1
-45	2
-46	1
-47	2
-48	1
-49	2
-50	1
-51	2
-52	1
-53	2
-54	1
-55	2
-56	1
-57	2
-58	1
-59	2
-60	1
-61	2
-62	1
-63	2
-64	1
-65	2
-66	1
-67	2
-68	1
-69	2
-70	1
-71	2
-72	1
-73	2
-74	1
-75	2
-76	1
-77	2
-78	1
-79	2
-80	1
-81	2
-82	1
-83	2
-84	1
-85	2
-86	1
-87	2
-88	1
-89	2
-90	1
-91	2
-92	1
-93	2
-94	1
-95	2
-96	1
-97	2
-98	1
-99	2

Home

Documentation

Run

Next Instruction

Reset Runtime

Clear Editor

Upload

Download

Examples

ASM Editor

```

35 BR L1
36 L0: NOOP
37 LOAD y
38 STORE T5
39 WRITE T5
40 STOP
41 T0 0
42 T1 0
43 T2 0
44 T3 0
45 T4 0
46 T5 0
47 x 0
48 y 0
49

```

Output:

Assembler Settings

Pop-out Documentation

Accumulator

-1

Variables

Variable	Value
T0	0
T1	0
T2	1
T3	2
T4	1
T5	-1
x	0

Stack

Top of the Stack is index 0

Position	Value
2	3
1	4
0	2
-1	1
-2	1
-3	2
-4	1
-5	2
-6	1
-7	2
-8	1
-9	2
-10	1
-11	2
-12	1
-13	2
-14	1
-15	2
-16	1
-17	2
-18	1
-19	2
-20	1
-21	2
-22	1
-23	2
-24	1
-25	2
-26	1
-27	2
-28	1
-29	2
-30	1
-31	2
-32	1
-33	2
-34	1
-35	2
-36	1
-37	2
-38	1
-39	2
-40	1
-41	2
-42	1
-43	2
-44	1
-45	2
-46	1
-47	2
-48	1
-49	2
-50	1
-51	2
-52	1
-53	2
-54	1
-55	2
-56	1
-57	2
-58	1
-59	2
-60	1
-61	2
-62	1
-63	2
-64	1
-65	2
-66	1
-67	2
-68	1
-69	2
-70	1
-71	2
-72	1
-73	2
-74	1
-75	2
-76	1
-77	2
-78	1
-79	2
-80	1
-81	2
-82	1
-83	2
-84	1
-85	2
-86	1
-87	2
-88	1
-89	2
-90	1
-91	2
-92	1
-93	2
-94	1
-95	2
-96	1
-97	2
-98	1
-99	2

Input 5

Output 5, 3, 1, 4, 2, 3, 1, 2, 1, -1