JAJ Language Reference

SER 502

Jyotinder singh, Ayush Gupta and jeme jhon

2016

Introduction:

JAJ is a procedural language execution just like C and execution begin from main. This parsing logic is done using JAVA and ANTLR4 which generate an intermediate code (.assm) which is supported by runtime written in python. This make JAJ platform independent.

Runtime:

The run is written in python called runtime.py. To execute <filename>.assm file user need to execute following command

Python runtime.py –f <filename>.assm

EIP instruction pointer

Execution of intermediate code start form setting the EIP MAIN\_START label this and then executing each line till we reach MAIN\_END. In between MAIN\_START there can we conditional statements (if-else), functional call, while loop, print statements, nested blocks and variable declaration (Global variable are also supported whose declaration is not between function.)

Scope of a variable can be global, local and within nested blocks. To manage this we have used global symbol table which take care of all the global variables, then to handle local variable a local call stack on which element creation happen to handle nested block when every a new block is created we create a new local scope and push the earlier local scope into stack. Now if a variable which is used in block which is not defined in this block we first search it parent then it parent and so on, after that we check the global scope.

Currently we are supporting only integer and Boolean but you can print strings using PRINT statement. Moreover we have STACK as a user defined variable. This user defined data structure can be manipulated using

TOP: will tell the top of the stack.

POP: will remove the top element.

PUSH: will push the element on the stack

EMPTY: will check if stack is empty or not.

To handle function call we use callstack. Whenever there is a CALL statement in the .assm file we save EIP+1 and local variable stack on call stack so after stack unwinding we can continue to execute sequentially. Moreover you can pass n number of arguments to the function by pushing all these argument on the argument stack.

We use general purpose register EAX to store intermediate result and when you want to return value from a function. R is another general purpose register which store the result of comparison CMP, which is then used by JNE (JUMP if not equal), JGE (JUMP if greater), JE (JUMP if equal) and JMP (is a unconditional JUMP)