SPSIL
Language Specification
Version 1.0

Dr. K. Muralikrishnan
kmurali@nitc.ac.in
NIT Calicut

March 3, 2012

Contents

1 Introduction

SPSIL or *System Programmer's Simple Integer Language* is an untyped programming language designed for implementation of an operating system on ESIM *(Extended Simple Integer Machine)* architecture. The language is minimilistic and consists of only basic constructs required for the implementation. Programming using SPSIL requires a basic understanding of the underlying ESIM architecture and operating system concepts.

2 Lexical Elements

2.1 Comments and White Spaces

SPSIL allows only line comments. Line comments start with the character sequence `//` and stop at the end of the line. White spaces in the program including tabs, newline and horizontal spaces are ignored.

2.2 Keywords

The following are the reserved words in SPSIL and it cannot be used as names of constatns.

```
alias    else      if     read   while
define   endif   ireturn  store
  do   endwhile    load    then
```

2.3 Operators and Delimiters

The following are the operators and delimiters in SPSIL

```
(   )   ;    [   ]   /   *   +   -   %
>   <   >=   <=  !=  ==  =   &&  ||  !
```

2.4 Registers

SPSIL allow the use of 19 registers for various operations.(R0-R15, BP, SP, IP)

2.5 Identifiers

Identifiers are used for defining names for constants as well as aliasing registers in the register set. Identifiers should start with an alphabet but may

contain both alphabets, digits and/or underscore (_). No other special characters are allowed in identifers.

2.6 Literals

There are integer literals and string literals in SPSIL. An integer literal is a sequence of digits representing an integer. Negative integers are represented with a negative sign preceding the sequence of digits. Any sequence of characters enclosed within double quotes (") are considered as string literals. However SPSIL restricts string literals to size of atmost 16 characters including the '\0' character which is implicitly appended at the end of a string value.

3 Register Set

SPSIL doesn't allow the use of declared variables. Instead a fixed set of registers is provided. The register set in SPSIL contains 19 registers. There is a direct mapping between the registers to the machine registers in ESIM.

R0-R7	Program Registers
R7-R15	Kernel Registers
BP	Base Pointer
SP	Stack Pointer
IP	Instruction Pointer

3.1 Aliasing

Registers in the register can be referred to by using a different name. A name is assigned to a particular register using the **alias** keyword. Each register can be assigned to only one alias at any particular point of time. However, a particular register can be assigned to a different alias at a later point. No two registers can have the same alias name simulatneously.

4 Named Constants

Symbolic names can be assigned to constants using the **define** keyword. Unlike aliasing, two or more symbolic names can be assigned to the same constant.

4.1 Predefined Constants

SPSIL provides a set of predefined named constants. These predefined names can also be reassigned to different constatns explicitly by the user. The predefined set of constants are mostly the starting addresses in memory for various OS components..

The predefined set of constants provided in SPSIL are

Name	Default Value
SCRATCHPAD	256
PAGE_TABLE	512
MEM_LIST	576
FILE_TABLE	640
READY_LIST	736
PROC_TABLE	767
FAT	1024
DISK_LIST	1536
USER_PROG	1792
INTERRUPT	13824

5 Addresses

The memory of the machine can be directly accessed. A word in the memory is accessed by specifying the word number within [ ]. A number, a register, or a constant can be used to specify the word number in the memory.

Examples of addressing : [1024], [R9], [FILE_TABLE], [CONSTANT]

6 Expressions

An expression specifies the computation of a value by applying operators and functions to operands. Function call in SPSIL are treated as expressions, and the value of the expression is its return value. SPSIL supports arithmetic and logical expressions

6.1 Arithmetic Expressions

Any integer value, variable, function returning an integer or 2 or more arithmetic expressions connected by arithmetic operators termed as arithmetic expressions. SPSIL provides five arithmetic operators, viz., +, -, *, / (Integer Division) and % (Modulo operator) through which arithmetic expressions

may be combined. Expression syntax and semantics are similar to standard practice in programming languages and normal rules of precedence, associativity and paranthesization hold. SPSIL is strongly typed, and hence the types of the oprands must match the operation.

6.2 Logical Expressions

Logical expressions may be formed by combining arithmetic expressions using relational operators. The relational operators supported by SPSIL are

```
<, >, <=, >=, ==, !=
```

Standard meanings apply to these operators. The operators take two arithmetic expressions as operands and the result will be a boolean value, either of 1(true) or 0(false). Only realational operator that can be applied to two strings is == (to check equality). This also considered as a Logical expression. Logical expressions themselves may be combined using logical operators, && (logical and) , || (logical or) and ! (not).

7 Statements

Statements control the execution of the program. All statements in SPSIL are terminated with a semicolon ;

7.1 Assignment Statement

The SPSIL assignment statement assigns the value of an expression to a variable, or an indexed array of the same type or a string value to a string variable. = is known as the assignment operator. Initialization during declaration is not allowed in SPSIL. The general syntax is as follows
variable_name = string_value / array_variable / expression

7.2 If Statement

If statements specify the conditional execution of two branches according to the value of a boolean expression. If the expression evaluates to true, the **if** branch is executed, otherwise, if present, the **else** branch is executed. The **else** part is optional. The general syntax is as follows

if *(logical expression)* ***then***
statements;

else
statements;
endif;

7.3 While Statement

While statement iteratively executes a set of statements based on a condition which is a logical expression. The statements are iteratively executed as long as the logical expression evaluates to true.

while *(logical expression)* ***do***
statements;
endwhile;

7.4 Return statement

Return statement in a function passes the control from the callee to the caller function and returns a value to the caller function. All functions including the **main()** must have exactly one **return** statement and it should be the last statement in the funtion body. The return type of the function should match the type of the expression. The return type of main is integer. The syntax is as follows

return *expression;*

7.5 Read/Write statements

The standard input and output statements in SPSIL are **read** and **write** respectively. The read statement reads an integer value from the standard input device into an integer variable or an indexed array variable or a string value into a string variable. The write statement outputs a string literal or the value of string variable or an arithmetic expression into the standard output.

read *variable_name;*
write *expression / string;*