SPSIL
Language Specification
Version 1.0

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

March 4, 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 minimalistic and consists only of 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 single line comments. 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 identifiers.

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

2.3 Operators and Delimiters

The following are the operators and delimiters in SPSIL

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

2.4 Registers

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

2.5 Identifiers

Identifiers are used as symbolic names for constants and aliases for registers. Identifiers should start with an alphabet but may contain alphabets, digits and/or underscore (_). No other special characters are allowed in identifiers.

2.6 Literals

Only integer literals are permitted 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.

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 these registers and the machine registers in ESIM.

R0-R7	Program Registers
R8-R15	Kernel Registers
BP	Base Pointer
SP	Stack Pointer
IP	Instruction Pointer

3.1 Aliasing

Registers 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 register can be reassigned to a different alias at a later point. No two registers can have the same alias name simultaneously.

4 Constants

Symbolic names can be assigned to values using the **define** keyword. Unlike aliasing, two or more names can be assigned to the same value.

4.1 Predefined Constants

SPSIL provides a set of predefined constants. These predefined constants can be assigned to different values 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. SPSIL supports arithmetic and logical expressions.

6.1 Arithmetic Expressions

Registers, constants, and 2 or more arithmetic operations connected using arithmetic operators are categorized 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.

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. A lrelational operator will take in two arguments and return 1 if the relation is valid and 0 otherwise. Logical expressions themselves may be combined using logical operators, && (logical and) , || (logical or) and ! (not).

6.3 String Comparison

The only operation that can be performed on strings stored in memory is string camparison. **strcmp** is used to compare two strings whose address is stored in the registers that are given as operands.

e.g. *strcmp(R9,R10);*

7 Addressing

SPSIL allows the use of directly addressing memory words of ESIM. A word in the memory is accessed by specifying the word number within [ ]. An arithmetic expression can be used to specify the word number. This will correspond to the value stored in the given address (word number). An expression can be assigned to an address. But an addressing element cannot be used in an arithmetic expression.

Examples of addressing: [1024], [R9], [R10+R11+128], [FAT] etc.

8 Statements

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

8.1 Define Statement

Define statement is used to define a symbolic name for a constant. Define statements should be the first block of an SPSIL program, i.e. it must be used before any other statment in an SPSIL program. The keyword **define** is used to associate a constant integer value to a name.

define *constant_name value;*

8.2 Alias Statement

An **alias** is used to refer to any register in the register set using a different name. **Alias** statements can be used anywhere in the program. Only the latest alias for any particular register can be used.

alias *alias_name register_name*

8.3 Assignment Statement

The SPSIL assignment statement assigns the value of an expression to a register or a memory address. textbf= is known as the assignment operator. The operand on the right hand side of the operator is assigned to the left hand side. The general syntax is as follows

Register / [Address] = Expression / [Address]

8.4 If Statement

If statements specify the conditional execution of two branches according to the value of a logical expression. If the expression evaluates to 1, the **if** branch is executed, otherwise the **else** branch is executed if it exists. The **else** part is optional. The general syntax is as follows

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

8.5 While Statement

While statement iteratively executes a set of statements based on the value of a logical expressions. The statements are iteratively executed as long as the logical expression evaluates to 1.

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

8.6 ireturn Statement

ireturn statement is used to pass control from kernel mode to user mode. The **ireturn** is used at the end of an interrupt code.

ireturn;

8.7 Read/Write Statements

The standard input and output statements in SPSIL are **read** and **write** respectively. The read statement reads an integer value into any register in the register set. The write statement outputs an integer, a string literal, the value inside a register, or a constant.

read *variable_name;*
write *expression;*