CPU-OS Simulator – Teaching Language

Syntax of simulator teaching language (STL)

```
program ProgName
    ... put program statement(s) here ...
end

library LibName
    ... put program statement(s) here ...
end
```

Examples of STL Statements

1. Conditional statements

```
if n <> 5 then
    ... put program statement(s) here ...
end if

if n > 0 and n < 6 then
    ... put program statement(s) here ...
end if

if n < 0 then
    ... put program statement(s) here ...
else
    ... put program statement(s) here ...
end if

select IdNo
    case 0
        ... put program case statement(s) here ...
    case 1
        ... put program case statement(s) here ...
    case else
        ... put program statement(s) here ...
end select

select IdNo
    case 0, 1, 2
        ... put program case statement(s) here ...
    case 10, 11
        ... put program case statement(s) here ...
    case else
        ... put program statement(s) here ...
end select
```
2. Loops

for n = 0 to 9
    ... put program statement(s) here ...
next

for n = 0 to 10 step 2
    ... put program statement(s) here ...
next

while n < 4
    ... put program statement(s) here ...
wend

do
    ... put program statement(s) here ...
loop

do
    ... put program statement(s) here ...
loop while n < 0

break

break * exits out of the outermost loop

continue

3. Subroutines

sub TestSub
    ... put program statement(s) here ...
end sub

fun TestAdd(param1, param2)
    Res = param1 + param2
    ... put more program statement(s) here ...
end sub = Res  -- return value is Res

sub TestSub intr 1
    ... put program statement(s) here ...
end sub
sub TestSub(ExpNo) exception
    ... put program statement(s) here ...
end sub

sub TestSub synchronise
    ... put program statement(s) here ...
end sub

sub TestSub as thread
    ... put program statement(s) here ...
end sub

sub TestSub as thread synchronise
    ... put program statement(s) here ...
end sub

Inlining subroutines:

inline sub TestSub
    ... put program statement(s) here ...
end sub

4. Variable declarations

For variables in memory
    var n integer
    var b boolean
    var s1 string
    var s2 string(20)
    var o object
    var a array(20) byte

For variables in registers
    regvar n integer
    regvar b boolean
    regvar s1 string
    regvar s2 string(20)
    regvar o object

5. Input output statements

    write (OutputData1, OutputData2, OutputData3, .....)

    writeln (OutputData1, OutputData2, OutputData3, .....)

Besim Mustafa - Aug 2014
writeln
read (InputData) - read is blocked
read (nowait, InputData) - read is not blocked

6. Calling subroutines

ACallVal = call Test1

call Test2(5, "Hello")
call Test3

or

ACallVal = Test1

Test2(5, "Hello")
Test3

7. Critical regions

Mutex:
enter - start of the critical region
leave - end of the critical region

counting semaphore:
semaphore(wait, sem) - sem is variable of type integer
semaphore(signal, sem)

8. Inbuilt libraries

getCPUNumber(CPUNo)
timer start 1000 - interrupt every 1000 msecs
timer stop
wait - used by parent thread waiting for children
wait(5) - time in seconds
date(DateStr)
allocmem(100, MemArry)  - allocates 100 bytes

9. Inter Process Communications (IPC)

  ipc (open, in, 2)
  ipc (receive, 2, PID, IPCMsg)
  ipc (open, out, 2)
  ipc (send, 3, 2, "Hello", block, timeout 3)
  ipc (close, 2)

10. Exception handling

guard
    ... put program statement(s) here ...
on exception(ExpNo) - ExpNo is returned after an exception
    ... put program statement(s) here ...
end guard

11. Resource allocate/deallocate

  resource(1, allocate)  - allocates resource number 1 - use resource numbers 0 to 5

  resource(2, free)  - frees resource number 2 - must have been previously allocated.

12. Assembler

  asm
    MOV #3, R01
    ADD R01, R02
  end

13. Compiler pragmas

  #PRAGMA UNROLLCNT = <number>  used by loop unrolling optimization - defines the maximum loop unrolling count.
14. Object-oriented features

class Class1
  var v1 integer is public
  var v2 string(10) is private

  sub Class1 is public
    … put program statement(s) here …
  end sub

  sub Class1(a, b) is public
    … put program statement(s) here …
  end sub

  sub Any1 (c) is public
    … put program statement(s) here …
  end sub

  sub ATest is private
    … put program statement(s) here …
  end sub

  … put class statement(s) here …
end

class Class2 inherits from Class1
  … put class statement(s) here …
end

var obj1 object
var obj2 object

instantiate obj1 as Class1

instantiate obj2 as Class2(“yes”, true)

call obj1.Any1(2)

call obj1.ATest

call obj2.Any1(7)