MINIPROJECT SUBPROGRAM CATALOG

# ***Source File: miniprojectUtils.asm***

1. **abort**
   1. **Used labels & registers**

*alertMessage: “Invalid input\n”*

*exitMessage: “Exitting…”*

* 1. **Purpose**

Pop up an abort string if input is invalid.

1. **appendStringRegister(%destination, %source)**
   1. **Used labels & registers**

*$v0: store character from a string after each loop*

*$a0: contains address of destination string*

*$a1: contains address of source string*

* 1. **Purpose**

Append %source string to %destination string

1. **appendLiteral(%label, %string)**
   1. **Used labels & registers**

*\_string\_: input string*

*$v0: load address of \_string\_*

* 1. **Purpose**

Append a literal string without label or register to a string label

1. **checkValidNotNegative(%value, %status)**
   1. **Used labels & registers**

*$t0: %value register*

*$t1: %status register*

*$t2: check register*

* 1. **Purpose**

check if %value is a non-negative number or empty

abort then exit program if %value is invalid and pass if %value is valid

1. **copyChar(%destination, %source)**
   1. **Used labels & registers**

*$t9: dummy register to contain input character*

* 1. **Purpose**

copy character from %source to %destination

1. **forLoop(%init, %cond, %increment, %body)**
   1. **Used labels & registers**

*$v0: contain result from %body value after each loop*

* 1. **Purpose**

simulate a for-loop as in C programming language

1. **getAddressAtIndex(%str, %index)**
   1. **Used labels & registers**

*$a0: address of %str label*

*$t0: %index to find*

* 1. **Purpose**

find address of a character in a string

1. **getCharDistance(%register\_1, %register\_2)**
   1. **Used labels & registers**

*$t0: %register\_1 contains a character 1*

*$t1: %register\_2 contains a character 2*

*$v0: abs($t0 - $t1)*

* 1. **Purpose**

find the difference in ASCII table of 2 registers, each of which contains a character

1. **getIntDialog(%label, %message)**
   1. **Used labels & registers**

*$a0: contains address of %mesage label*

*$a1: contains status after calling syscall 51*

*$v0: used for syscall 51*

* 1. **Purpose**

get input integer value from a pop-up dialog and then save it to %label if valid

1. **getLengthRegister(%register)**
   1. **Used labels & registers**

*$a0: contains address of %register*

*$t1: contains address of a character in a string in each loop*

*$t2: character in each loop*

*$v0: contains output length*

* 1. **Purpose**

find the length of a string, whose address is stored in a register

1. **getLengthLabel(%label)**
   1. **Used labels & registers**

*$a0: contains address of %label*

*$v0: contains output length*

* 1. **Purpose**

find the length of a string stored in a label

1. **getOrdinal(%address)**
   1. **Used labels & registers**

*$v0: contains character loaded from %address*

* 1. **Purpose**

find ASCII value of a character

1. **getStringDialog(%label)**
   1. **Used labels & registers**

*message: “Please input a string”*

*$v0: used for syscall 54*

*$a0: contains address of message*

*$a1: contains address of %label – input string*

*$a2: maximum length of a string*

* 1. **Purpose**

get input string value from a pop-up dialog and then save it to %label

1. **getStringIndex(%string, %register\_index)**
   1. **Used labels & registers**

*$t0: contains value from %register\_index*

*$t1: contains address of %string*

*$v0: contains character found*

* 1. **Purpose**

get character from an input string and index to find

1. **increaseBy1(%label)**
   1. **Used labels & registers**

*$t9: contains value from %label*

* 1. **Purpose**

increase label value by 1

1. **popRegister(%register)**
   1. **Used labels & registers**

*$sp: stack pointer*

* 1. **Purpose**

pop from stack and save the popped value to %register

1. **printIntLabel(%label)**
   1. **Used labels & registers**

*$v0: used for syscall 1*

*$a0: contains value from %label*

* 1. **Purpose**

print integer from a label to console

1. **printIntRegister(%register)**
   1. **Used labels & registers**

*$v0: used for syscall 1*

*$a0: contains value from %register*

* 1. **Purpose**

print integer from a register to console

1. **printStringLabel(%label)**
   1. **Used labels & registers**

*$v0: used for syscall 4*

*$a0: contains value from %label*

* 1. **Purpose**

print string from a label to console

1. **printStringRegister(%register)**
   1. **Used labels & registers**

*$v0: used for syscall 4*

*$a0: contains value from %register*

* 1. **Purpose**

print string from a register to console

1. **printLiteral(%string)**
   1. **Used labels & registers**

*$v0: used for syscall 4*

*$a0: contains value from %string*

* 1. **Purpose**

print a literal string to console

1. **printStringIntDialog(%label\_string, %label\_int)**
   1. **Used labels & registers**

*$v0: used for syscall 56*

*$a0: contains address of %label\_string*

*$a1: contains value of %label\_int*

* 1. **Purpose**

print a string followed by an integer inside a pop-up dialog

1. **printStringStringDialog(%label\_stringA, %label\_stringB)**
   1. **Used labels & registers**

*$v0: used for syscall 59*

*$a0: contains address of %label\_string*

*$a1: contains address of %label\_string*

* 1. **Purpose**

print a string followed by another string inside a pop-up dialog

1. **pushRegister(%register)**
   1. **Used labels & registers**

*$sp: stack pointer*

* 1. **Purpose**

push a register into stack