printReg

Description: print the value in the RDI register

Preconditions: the value to be displayed must be in rdi

Postconditions: the value in rdi is displayed in the format 0xAABBCCDDEEFF1122

Registers rax, rbx, rcx, rdx, rsi and rdi are unchanged after the call

example:

```
; Data
section  .data
extern printReg
extern exitNormal

; Code
section      .text

global _start

_start:

  mov rdi, 0x1234567890abcd
  call printReg
  call exitNormal
```

output:

```
0x1234567890ABCD
```

printRAX

Description: print the value in the RAX register

Preconditions: the value to be displayed must be in rax

Postconditions: the value in rax is displayed in the format 0xAABBCCDDEEFF1122

Registers rax, rbx, rcx, rdx, rsi and rdi are unchanged after the call
example:
```
; Data
section  .data
extern printRAX
extern exitNormal

; Code
section      .text

global _start

_start:
   mov rax, 0x1234567890ABCD
   call printRAX
   call exitNormal
output:
0x1234567890ABCD
```

printRBX

Description: print the value in the RBX register

Preconditions: the value to be displayed must be in rbx

Postconditions: the value in rbx is displayed in the format 0xAABBCCDDEEFF1122

Registers rax, rbx, rcx, rdx, rsi and rdi are unchanged after the call
example:

```
; Data
section  .data
extern printRBX
extern exitNormal

; Code
section      .text

global _start

_start:

   mov rbx, 0x1234567890ABCD
   call printRBX
   call exitNormal
```
output:
```
0x1234567890ABCD
```

printRCX

Description: print the value in the RCX register

Preconditions: the value to be displayed must be in rcx

Postconditions: the value in rcx is displayed in the format 0xAABBCCDDEEFF1122

Registers rax, rbx, rcx, rdx, rsi and rdi are unchanged after the call
example:
```
; Data
section  .data
extern printRCX
extern exitNormal

; Code
section      .text
```

```
global _start

_start:

   mov rcx, 0x1234567890ABCD
   call printRCX
   call exitNormal
```
output:
```
0x1234567890ABCD
```

printRDX

Description: print the value in the RDX register

Preconditions: the value to be displayed must be in rdx

Postconditions: the value in rdx is displayed in the format 0xAABBCCDDEEFF1122

Registers rax, rbx, rcx, rdx, rsi and rdi are unchanged after the call

example:
```
; Data
section  .data
extern printRDX
extern exitNormal

; Code
section      .text

global _start

_start:

   mov rdx, 0x1234567890ABCD
   call printRDX
   call exitNormal
```
output:
```
0x1234567890ABCD:w
```

printABCD

Description: print the value in the RAX, RBX, RAX and RDX registers

Preconditions: the value to be displayed must be in rax, rbx, rcx and rdx registers

Postconditions: the value in rax, rbx, rcx and rdx are displayed in the format 0xAABBCCDDEEFF1122

a endline is printed after each register is printed

Registers rax, rbx, rcx, rdx, rsi and rdi are unchanged after the call

example:
```
; Data
section  .data
extern printABCD
extern exitNormal

; Code
section      .text

global _start

_start:
   mov rax, 0xAAAA
   mov rbx, 0xBBBB
   mov rcx, 0xCCCC
   mov rdx, 0xDDDD
   call printABCD
   call exitNormal
```
output:
```
0x000000000000AAAA
0x000000000000BBBB
0x000000000000CCCC
0x000000000000DDDD
```

printMSG

Description: print the message associated with the value in RDI

Preconditions: the value to be displayed must be in rdi

Postconditions: the message cooresponding to the value in rdi is displayed as indicated below

Registers rax, rbx, rcx, rdx, rsi and rdi are unchanged after the call

0x0 0XA (ENDL)

0x1 MOV

0x2 ADD

0x3 SUB

0x4 MUL

0x5 DIV

0x6 Signed

0x7 Unsigned

0x8 ' ' (SPACE)

0x9 RAX

0xA RBX

0xB RCX

0xC RDX

0xD CS12

0xE AND

0xF OR

0x10 XOR

0x11 NOT

0x12 SHIFT

0x13 ROTATE

0x14 LEFT

0x15 RIGHT

0x15 "Enter up to a quadword in hex: example:ABCDEF1234567890"

example:

```
; Data
section  .data
extern printMSG
extern exitNormal

; Code
section      .text
```

```
global _start

_start:

   mov rdi, 0x1
   call printMSG
   call exitNormal
```
output: (No Carriage Return / Line Feed)
```
MOV
```

printEndl

Description: print the endline character

Preconditions: None

Postconditions: an endline is printed

Registers rax, rbx, rcx, rdx, rsi and rdi are unchanged after the call

example:
```
; Data
section  .data
extern printEndl
extern exitNormal

; Code
section      .text

global _start

_start:

   call printEndl
   call exitNormal
```
output: (A blank Line)

printSpace

Description: print the space character

Preconditions: None

Postconditions: a space is printed

Registers rax, rbx, rcx, rdx, rsi and rdi are unchanged after the call

example:
```
; Data
section  .data
extern printSpace
extern exitNormal

; Code
section      .text

global _start

_start:

   call printSpace
   call exitNormal
```
output: (a space, but no return)

getQuad

Description: get a Quad Word from the user and put the result in RAX

the user will type in characters 0-9,a-f,A-F.

Preconditions: None

Postconditions: rax contains the value entered by the user up to 16 characters translated into hex from ASCII

Registers rbx, rcx, rdx, rsi and rdi are unchanged after the call

rax contains the value input by the user translated into a quad word

```
; Data
section  .data
extern printMSG
extern printRAX
extern printEndl
extern getQuad
extern exitNormal

; Code
section      .text

global _start
_start:
     ; output message to user to input a Quad Word
     mov rdi, 0x16
     call printMSG
     call printEndl    ; endline

     ; get a 16byte entry from the user that represents a 
quadword
     call getQuad
     call printRAX ; print the result

     call exitNormal
```

output:

```
Enter up to a quadword in hex: example:ABCDEF12345678
```

```
123456abcd
0x000000123456ABCD
```

getByteArray

Description: bytes are placed in memory starting at the address pointed to by the RSI Register

Preconditions: a byte array must exist large enough to hold the values input by the user

rsi must point to the address of the byte array to fill

rdx must contain the value of the number of characters to read into the byte buffer

Postconditions: The byte array pointed to by the rdi will contain the characters input by the user in ASCII

Registers rax, rbx, rcx, rdx, rsi and rdi are unchanged after the call

```
; Data
section  .data
extern getByteArray
extern printByteArray
extern printEndl
extern exitNormal
array        db   "Input 16 bytes  "
numberOfBytesdq   0x10


; Code
section      .text

global _start
_start:
    ; printByteArray
    mov rsi, array            ; note moving the address 
not the value
    mov rdx, [numberOfBytes]  ; print this many bytes of 
the array, value not address
    call printByteArray       ; print the array
    call printEndl

    ; getByteArray
    mov rsi, array            ; note moving the address 
not the value
    mov rdx, [numberOfBytes]  ; get this many bytes of 
the array, value not address
    call getByteArray     ; get the array

    ; printByteArray
    mov rsi, array            ; note moving the address 
not the value
    mov rdx, [numberOfBytes]  ; print this many bytes of 
the array, value not address
    call printByteArray       ; print the array
    call printEndl

    call exitNormal
Input 16 bytes
abcdefghijklmnop
abcdefghijklmnop
```

printByteArray

Description: bytes are read from the memory address pointed to by the RSI Register and output to stdout

Preconditions: a byte array must exist with the desired output

rsi must point to the address of the byte array to read

rdx must contain the value of the number of characters to write to stdout

Postconditions: The byte array pointed to by the rdi will have been printed to stdout

Registers rax, rbx, rcx, rdx, rsi and rdi are unchanged after the call

```
; Data
section  .data
extern printByteArray
extern printEndl
extern exitNormal
arrayToPrint db   "Print This Array"
numberOfBytes dq   0x10

; Code
section      .text

global _start
_start:
    ; printByteArray
    mov rsi, arrayToPrint           ; note moving the
address not the value
    mov rdx, [numberOfBytes]   ; print this many bytes of
the array, value not address
    call printByteArray       ; print the array

    call printEndl
    call exitNormal
```

output:

```
Print This Array
```

exitNormal

Description: Exit to Linux with returning 0

Preconditions: None

Postconditions: A 0 is returned and an exit executed, returning control to the operating system
example:
```
; Data
section  .data
extern exitNormal

; Code
section      .text

global _start

_start:

   call exitNormal
```
output: (None, but you will not get a Segmentation fault)

getRand

Description: Get a pseudorandom number and place it in RAX

Preconditions: None

Postconditions: A pseudorandom number is in RAX (This is actually just the system clock, so not really random)
example:
```
; Data
section  .data
extern getRand

; Code
```

```
section      .text

global _start

_start:
   call  getRand
   call  printRAX
   call  exitNormal
```