CPSC-240 Computer Organization and Assembly Language

Chapter 13

System Services

Instructor: Yitsen Ku, Ph.D.
Department of Computer Science,
California State University, Fullerton, USA

CALIFORNIA STATE UNIVERSITY FULLERTON™

CALIFORNIA STATE UNIVERSITY FULLERTON

Outline

- Calling System Services
- Newline Character
- Console Output
- Console Input
- File Open Operations
- File Read
- File Write
- File Operation Examples

CALIFORNIA STATE UNIVERSITY FULLERTON

Calling System Services

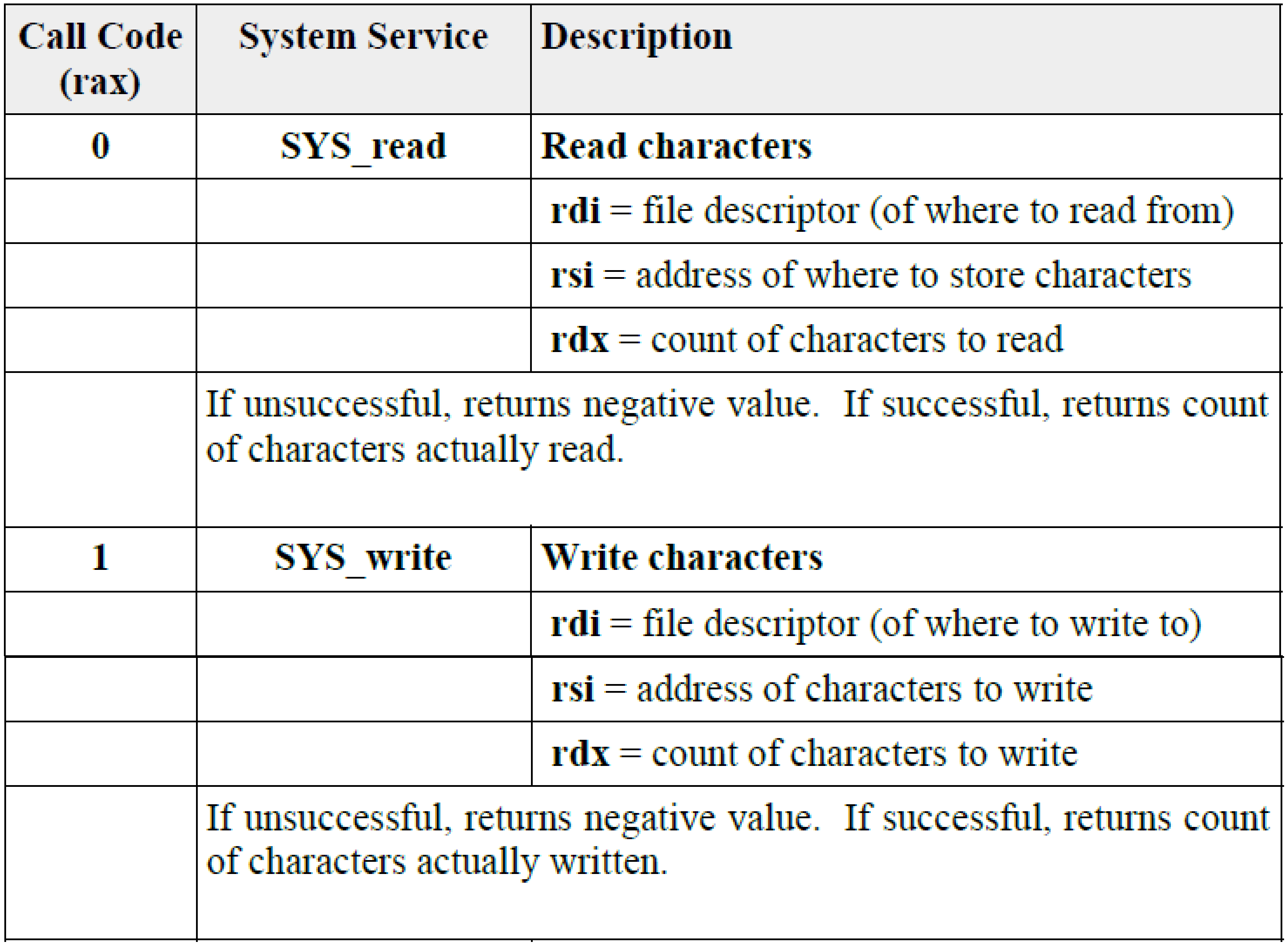
Calling System Services

- A system service call is logically similar to calling a function, where the function code is located within the operating system.
- When calling system services, arguments are placed in the standard argument registers.
- System services do not typically use stack-based arguments.
- This limits the arguments of a system services to six (6), which does not present a significant limitation.

Argument Locations and Call Convention

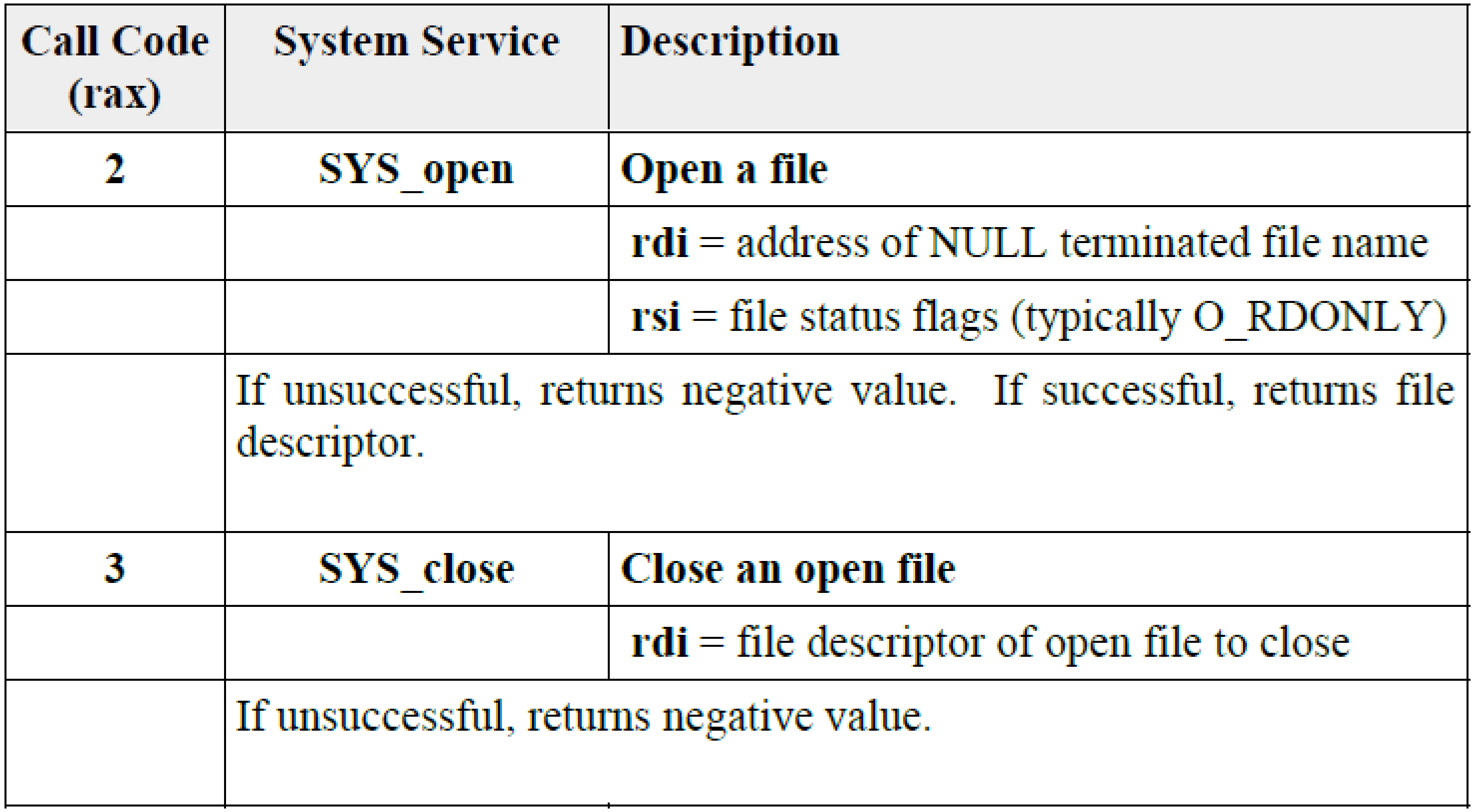
Register	Usage
`rax`	Call code (see table)
`rdi`	1st argument (if needed)
`rsi`	2nd argument (if needed)
`rdx`	3rd argument (if needed)
`r10`	4th argument (if needed)
`r8`	5th argument (if needed)
`r9`	6th argument (if needed)

SYS_read and SYS_write



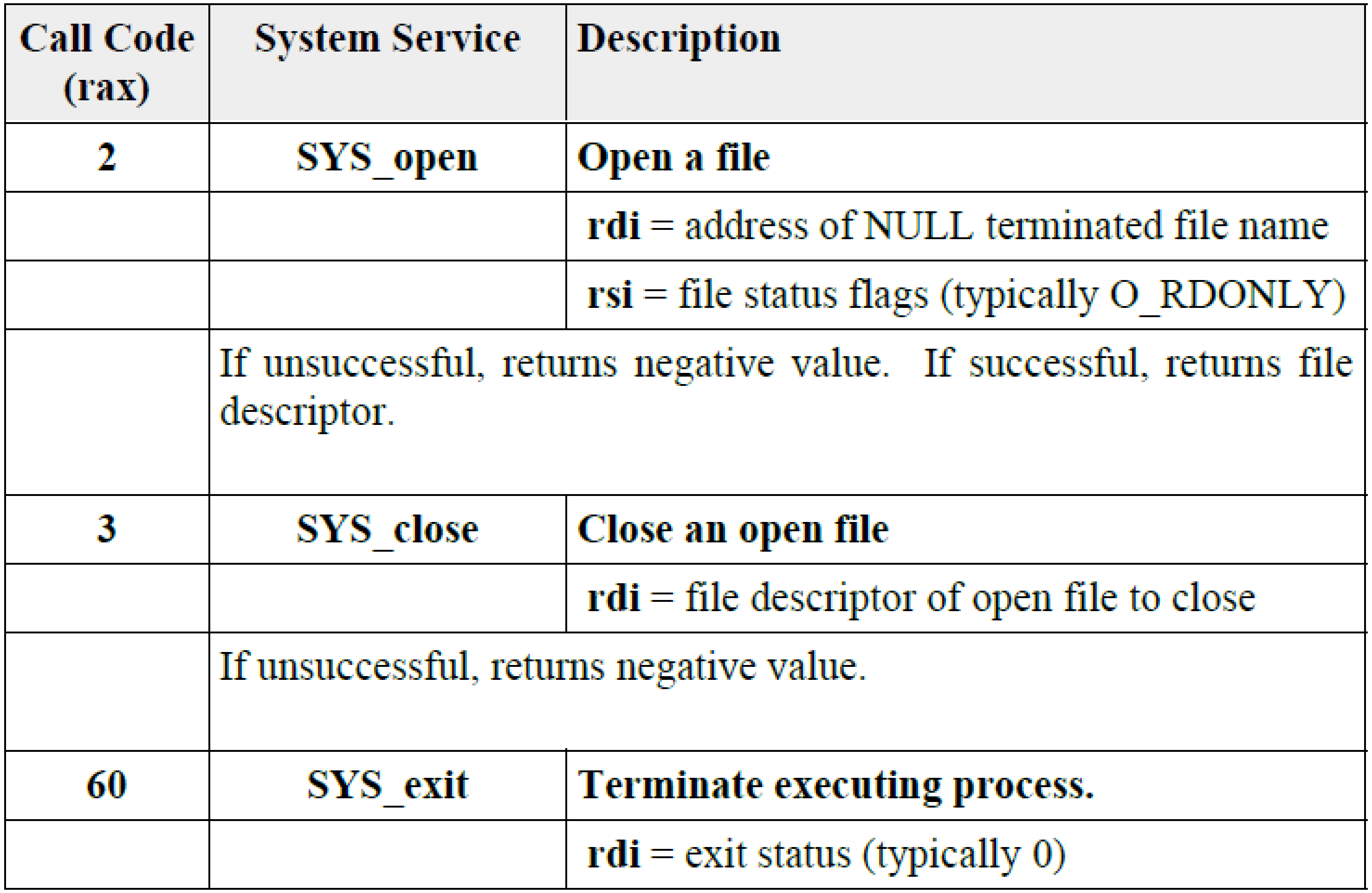
Call Code (rax)	System Service	Description
0	**SYS_read**	**Read characters**
		rdi = file descriptor (of where to read from)
		rsi = address of where to store characters
		rdx = count of characters to read
	If unsuccessful, returns negative value. If successful, returns count of characters actually read.	
1	**SYS_write**	**Write characters**
		rdi = file descriptor (of where to write to)
		rsi = address of characters to write
		rdx = count of characters to write
	If unsuccessful, returns negative value. If successful, returns count of characters actually written.	

SYS_open and SYS_close



Call Code (rax)	System Service	Description
2	**SYS_open**	**Open a file**
		rdi = address of NULL terminated file name
		rsi = file status flags (typically O_RDONLY)
	If unsuccessful, returns negative value. If successful, returns file descriptor.	
3	**SYS_close**	**Close an open file**
		rdi = file descriptor of open file to close
	If unsuccessful, returns negative value.	

SYS_open, SYS_close, and SYS_exit



Call Code (rax)	System Service	Description
2	**SYS_open**	**Open a file**
		rdi = address of NULL terminated file name
		rsi = file status flags (typically O_RDONLY)
	If unsuccessful, returns negative value. If successful, returns file descriptor.	
3	**SYS_close**	**Close an open file**
		rdi = file descriptor of open file to close
	If unsuccessful, returns negative value.	
60	**SYS_exit**	**Terminate executing process.**
		rdi = exit status (typically 0)

Newline Character

Newline Character

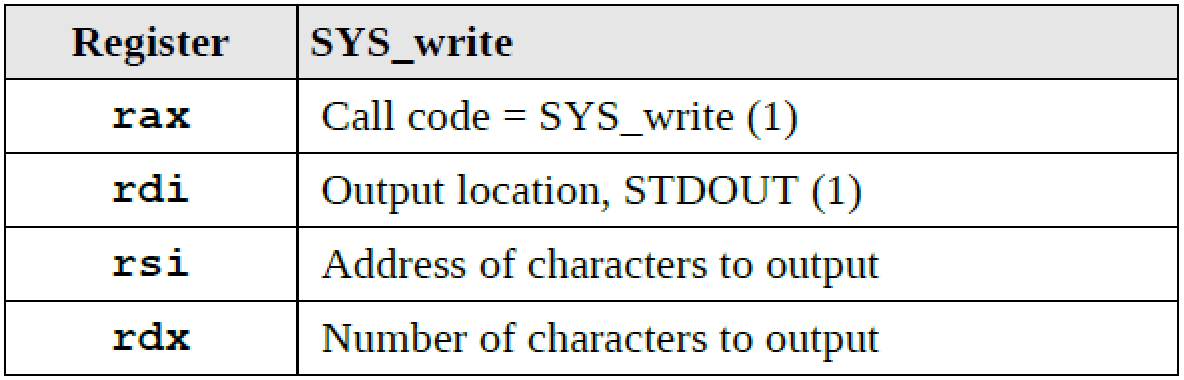
- In Unix/Linux systems, the linefeed, abbreviated LF with an ASCII value of 10 (or 0x0A), is used as the newline character.
- In Windows systems, the newline is carriage return, abbreviated as CR with an ASCII value 13 (or 0x0D) followed by the LF.
- The LF is used in the code examples in the text.

CALIFORNIA STATE UNIVERSITY FULLERTON

Console Output

Console Output

- The system service to output characters to the console is the system write (SYS_write).
- The arguments for the write system service are as follows:



Register	SYS_write
`rax`	Call code = SYS_write (1)
`rdi`	Output location, STDOUT (1)
`rsi`	Address of characters to output
`rdx`	Number of characters to output

Console Output

- Assuming the following declarations:

```
STDOUT     equu     1                  ; standard output
```

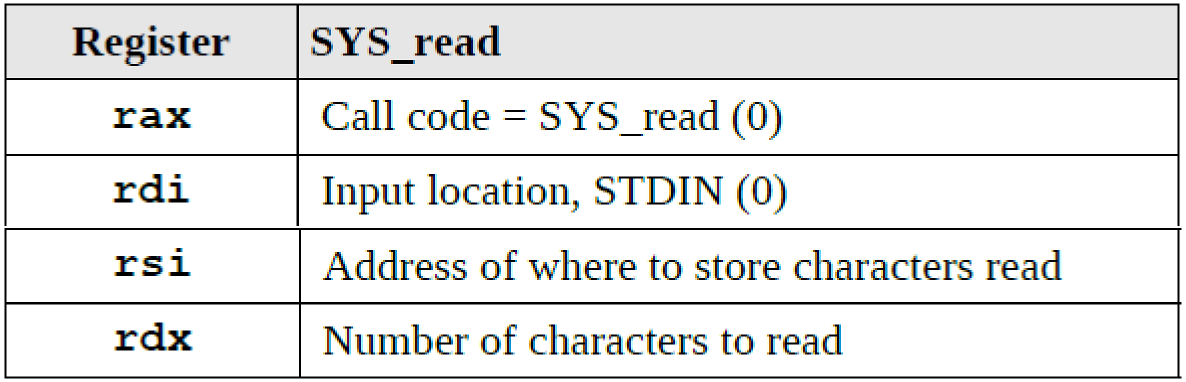
- For example, to output "Hello World" (it's traditional) to the console, the system write (SYS_write) would be used. The code would be as follows:

```
mov        rax, SYS_write              ; rax = 1
mov        rdi, STDOUT                 ; rdi = 1
mov        rsi, msg                    ; rsi = msg address
mov        rdx, qword [msgLen]         ; rdx = 11
syscall
```

Console Input

Console Input

- The system service to read characters from the console is the system read (SYS_read).
- SYS_read will read one character at a time until a LF (the Enter key) is read. Each character will be read and then stored, one at a time, in an appropriately sized array.
- The arguments for the read system service are as follows:



Register	SYS_read
`rax`	Call code = SYS_read (0)
`rdi`	Input location, STDIN (0)
`rsi`	Address of where to store characters read
`rdx`	Number of characters to read

CALIFORNIA STATE UNIVERSITY FULLERTON

Console Input

- Assuming the following declarations:

```
STDIN       equu    0                       ; standard input
SYS_read    equ     0                       ; call code for read

inChar      db      0                       ; [inChar] = 0
```

- For example, to read a single character from the keyboard, the system read (SYS_read) would be used. The code would be as follows:

```
mov         rax, SYS_read                   ; rax = 0
mov         rdi, STDIN                      ; rdi = 0
mov         rsi, inChar                     ; rsi = inChar address
mov         rdx, 1                          ; rdx = 1
syscall
```

One-Digit ASCII Addition

```
asc1    db      '2'                     ;asc1 = 32h
asc2    db      '5'                     ;asc2 = 35h
sum     db      '0'                     ;sum = 30h     .
.
mov     al, byte[asc1]                  ;al = asc1 = 32h
add     al, byte[asc2]                  ;al = al + asc2 = 67h
mov     ah, 0                           ;ah = 0
and     al, 0fh                         ;al = 67h & 0fh = 07h
add     byte[sum], al                   ;sum = al = 37h = '7'
```

One-Digit ASCII Subtraction

```
asc1    db    '9'                ;asc1 = 39h
asc2    db    '4'                ;asc2 = 34h
dif     db    '0'                ;dif = 30h      .
.
mov     al, byte[asc1]           ;al = asc1 = 39h
sub     al, byte[asc2]           ;al = al - asc2 = 05h
add     byte[dif], al            ;dif = 30h + 05h = 35h = '5'
```

One-Digit ASCII Multiplication

```
asc1    db      '6'                     ;asc1 = 36h
asc2    db      '3'                     ;asc2 = 33h
prod    db      '00'                    ;prod = 3030h  .
.
mov     al, byte[asc1]                  ;al = asc1 = 36h
and     al, 0fh                         ;al = 36h & 0fh = 06h
mov     bl, byte[asc2]                  ;bl = asc2 = 33h
and     bl, 0fh                         ;bl = 33h & 0fh = 03h
mul     bl                              ;ax = al*bl = 12h
mov     ah, 0                           ;ah = 0
mov     cl, 10                          ;cl = 10 = 0ah
div     cl                              ;ah=ax mod 10, al=ax/10
add     byte[prod+0], al                ;[prod+0]=30h+01h=31h
add     byte[prod+1], ah                ;[prod+1]=30h+08h=38h
```

CALIFORNIA STATE UNIVERSITY FULLERTON

One-Digit ASCII Division

```
asc1    db      '18'                        ;asc1 = 3138h
asc2    db      '3'                         ;asc2 = 33h
quan    db      '0'                         ;quan = 30h
remd    db      '0'                         ;remd = 30h
.
and     byte[asc1+0], 0fh                   ;asc1+0 = 01h
and     byte[asc1+1], 0fh                   ;asc1+1 = 08h
and     byte[asc2], 0fh                     ;asc2 = 03h
mov     al, byte[asc1+0]                    ;al = 01h
mov     bl, 10                              ;bl = 10 = 0ah
mul     bl                                  ;ax = al*bl = 1*10 = 10
add     al, byte[asc1+1]                    ;al = al+08h = 10+8 = 18 = 12h
div     byte[asc2]                          ;al=ax/asc2=06h, ah=ax%asc2=0
add     quan, al                            ;quan = 36h
add     remd, ah                            ;remd = 30h
```

File Open Operations

File Open

- The file open requires that the file exists in order to be opened. If the file does not exist, it is an error.
- The file open operation also requires the parameter flag to specify the access mode.
- The access mode must include one of the following:
  - Read-Only Access → O_RDONLY
  - Write-Only Access → O_WRONLY
  - Read/Write Access → O_RDWR

File Open

- The arguments for the file open system service are as follows:

Register	SYS_open
`rax`	Call code = SYS_open (2)
`rdi`	Address of NULL terminated file name string
`rsi`	File access mode flag

- Assuming the following declarations:

```
SYS_open     equu    2          ; file open
O_RDONLY     equ     000000q    ; read only
O_WRONLY     equ     000001q    ; write only
O_RDWR       equ     000002q    ; read and write
```

File Open/Create

- A file open/create operation will create a file. If the file does not exist, a new file will be created. If the file already exists, it will be erased and a new file created. Thus, the previous contents of the file will be lost.
- The arguments for the file open/create system service are as follows:

Register	SYS_creat
`rax`	Call code = SYS_creat (85)
`rdi`	Address of NULL terminated file name string
`rsi`	File access mode flag

File Open/Create

- Assuming the following declarations:

```
SYS_creat   equ   85        ; file open
O_CREAT     equ   0x40
O_TRUNC     equ   0x200
O_APPEND    equ   0x400
S_IRUSR     equ   00400q    ; owner, read permission
S_IWUSR     equ   00200q    ; owner, write permission
S_IXUSR     equ   00100q    ; owner, execute permission
```

- The file status flags "S_IRUSR | S_IWUSR" would allow simultaneous read and write, which is typical. The "|" is a logical OR operation, thus combining the selections.
- If the file open/create operation does not succeed, a negative value is returned in the **rax** register. If file open/create operation succeeds, a file descriptor is returned.

File Write

File Write

- The arguments for the file write system service are as follows:

Register	SYS_write
`rax`	Call code = SYS_write (1)
`rdi`	File descriptor (of open file)
`rsi`	Address of characters to write
`rdx`	Count of characters to write

- Assuming the following declarations:
  SYS_write equ 0 ; file write
- If the file write operation does not succeed, a negative value is returned in the **rax** register. If the file write operation does succeed, the number of characters actually written is returned.

File Write Example (1)

```
section     .data
LF          equu    10                    ; line feed
NULL        equ     0                     ; end of string
SYS_write   equ     1                     ; write
fileName    db      "url.txt", NULL
url         db      "http://www.google.com"
            db      LF, NULL
len         dq      $-url-1
writeDone   db      "Write Completed.", LF, NULL
fileDesc    dq      0
errMsgOpen  db      "Error opening file.", LF, NULL
errMsgWrite db      "Error writing to file.", LF, NULL
section     .text
global _start
_start:
; Attempt to open file.
; Use system service for file open
; System Service - Open/Create
; rax = SYS_creat (file open/create)
; rdi = address of file name string
; rsi = attributes (i.e., read only, etc.)
```

CALIFORNIA STATE UNIVERSITY FULLERTON

File Write Example (2)

```
openInputFile:
    mov     rax, SYS_creat                  ; file open/create
    mov     rdi, fileName                   ; file name string
    mov     rsi, S_IRUSR | S_IWUSR          ; allow read/write
    syscall                                 ; call the kernel
    cmp     rax, 0                          ; check for success
    jl      errorOnOpen
    mov     qword [fileDesc], rax           ; save descriptor

; -----
; System Service - write
; rax = SYS_write
; rdi = file descriptor
; rsi = address of characters to write
; rdx = count of characters to write
; Returns:
; if error -> rax < 0
; if success -> rax = count of characters actually read
; Write to file.
```

File Write Example (3)

```
    mov     rax, SYS_write
    mov     rdi, qword [fileDesc]
    mov     rsi, url
    mov     rdx, qword [len]
    syscall
    cmp     rax, 0
    jl      errorOnWrite
    mov     rdi, writeDone
    call    printString
; -----
; Close the file.
; System Service - close
; rax = SYS_close
; rdi = file descriptor
    mov     rax, SYS_close
    mov     rdi, qword [fileDesc]
    syscall
    jmp     exampleDone
```

CALIFORNIA STATE UNIVERSITY FULLERTON

File Write Example (4)

```
; ----- Error on open.
; note, rax contains an error code which is not used
; for this example.
errorOnOpen:
    mov         rdi, errMsgOpen
    call        printString
    jmp         exampleDone

; ----- Error on write.
; note, rax contains an error code which is not used
; for this example.
errorOnWrite:
    mov         rdi, errMsgWrite
    call        printString
    jmp         exampleDone

; ----- Example program done.
exampleDone:
    mov         rax, SYS_exit
    mov         rdi, EXIT_SUCCESS
    syscall
```

CALIFORNIA STATE UNIVERSITY FULLERTON

File Write Example (5)

```
; ***********************************************************
; Generic function to display a string to the screen.
; String must be NULL terminated.
; Algorithm:
;     Count characters in string (excluding NULL)
;     Use syscall to output characters
; Arguments:
;     1) address, string
; Returns: nothing

global printString                              ; external function
printString:
    .
    .
    .
```

CALIFORNIA STATE UNIVERSITY FULLERTON

File Read

File Open/Create

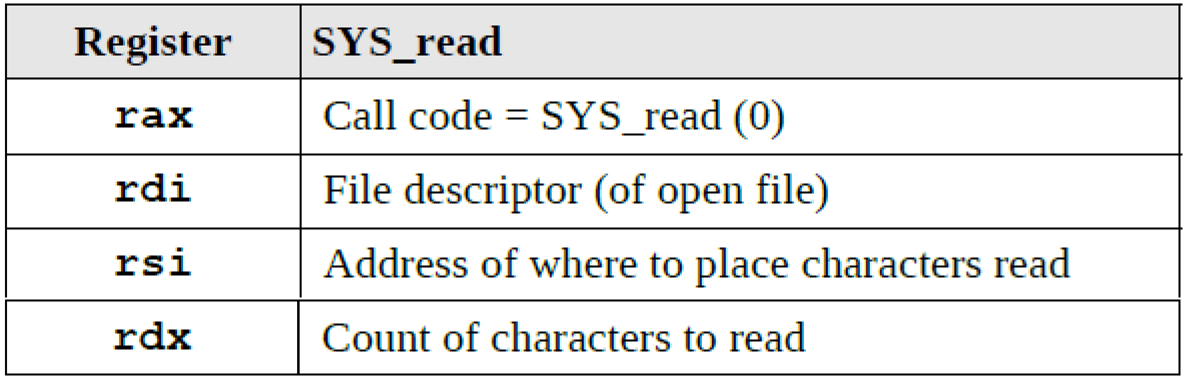
- Assuming the following declarations:

```
SYS_creat   equ   85          ; file open
O_CREAT     equ   0x40
O_TRUNC     equ   0x200
O_APPEND    equ   0x400
S_IRUSR     equ   00400q      ; owner, read permission
S_IWUSR     equ   00200q      ; owner, write permission
S_IXUSR     equ   00100q      ; owner, execute permission
```

- The file status flags "S_IRUSR | S_IWUSR" would allow simultaneous read and write, which is typical. The "|" is a logical OR operation, thus combining the selections.
- If the file open/create operation does not succeed, a negative value is returned in the **rax** register. If file open/create operation succeeds, a file descriptor is returned.

File Read

- A file must be opened with the appropriate file access flags before it can be read. The arguments for the file read system service are as follows:



Register	SYS_read
`rax`	Call code = SYS_read (0)
`rdi`	File descriptor (of open file)
`rsi`	Address of where to place characters read
`rdx`	Count of characters to read

- Assuming the following declarations:
  SYS_read equ 0 ; file read
- If the file read operation does not succeed, a negative value is returned in the **rax** register. If the file read operation succeeds, the number of characters actually read is returned.

File Read Example (1)

```
section      .data
LF           equu    10                               ; line feed
NULL         equ     0                                ; end of string
SYS_write    equ     1                                ; write
BUFF_SIZE    equ     255
fileName     db      "url.txt", NULL
fileDesc     dq      0
errMsgOpen   db      "Error opening the file.", LF, NULL
errMsgRead   db      "Error reading from the file.", LF, NULL

section      .bss
readBuffer   resb    BUFF_SIZE

section      .text
global _start
_start:
; Attempt to open file.
; Use system service for file open
; System Service - Open
; rax = SYS_open (file open)
; rdi = address of file name string
; rsi = attributes (i.e., read only, etc.)
```

File Read Example (2)

```
openInputFile:
    mov         rax, SYS_open                          ; file open
    mov         rdi, fileName                          ; file name string
    mov         rsi, O_RDONLY                          ; read only access
    syscall                                            ; call the kernel
    cmp         rax, 0                                 ; check for success
    jl          errorOnOpen
    mov         qword [fileDesc], rax                  ; save descriptor

; -----
; System Service - read
; rax = SYS_read
; rdi = file descriptor
; rsi = address of where to place data
; rdx = count of characters to read
; Returns:
; if error -> rax < 0
; if success -> rax = count of characters actually read
```

File Read Example (3)

```
    mov     rax, SYS_read
    mov     rdi, qword [fileDesc]
    mov     rsi, readBuffer
    mov     rdx, BUFF_SIZE
    syscall
    cmp     rax, 0
    jl      errorOnRead
; ----- Print the buffer.
; add the NULL for the print string
    mov     rsi, readBuffer
    mov     byte [rsi+rax], NULL
    mov     rdi, readBuffer
    call    printString
; ----- Close the file.
; System Service - close
; rax = SYS_close
; rdi = file descriptor
    mov     rax, SYS_close
    mov     rdi, qword [fileDesc]
    syscall
    jmp     exampleDone
```

File Read Example (4)

```
; ----- Error on open.
; note, rax contains an error code which is not used
; for this example.
errorOnOpen:
    mov         rdi, errMsgOpen
    call        printString
    jmp         exampleDone

; ----- Error on write.
; note, rax contains an error code which is not used
; for this example.
errorOnWrite:
    mov         rdi, errMsgWrite
    call        printString
    jmp         exampleDone

; ----- Example program done.
exampleDone:
    mov         rax, SYS_exit
    mov         rdi, EXIT_SUCCESS
    syscall
```

File Read Example (5)

```
; **********************************************************
; Generic function to display a string to the screen.
; String must be NULL terminated.
; Algorithm:
;     Count characters in string (excluding NULL)
;     Use syscall to output characters
; Arguments:
;     1) address, string
; Returns: nothing

global printString                              ; external function
printString:
    .
    .
    .
```

Thanks