Assembly – Linux x86

x86 Registers

EAX – Accumulator, optimised for numeric operations, i.e. arithmetic, system calls and return values.

EBX - Base register, often used for base pointer in memory Addressing.

ECX - Counter, commonly used in loop iterations and string operations.

EDX - Data register, used in multiplication, division, and I/O.

ESI - Source Index, used for source pointer in string and memory Operations.

EDI - Destination Index, used for destination pointer in string and memory operations.

ESP - Stack Pointer, tracks the top of the stack.

EBP - Base Pointer, maintains the base of the current stack frame.

Linux Interrupts

0x00: Divide by zero error

0x01: Debug exception

0x02: Non-maskable interrupt (NMI)

0x03: Breakpoint

0x04: Overflow

0x05: Bounds check

0x06: Invalid opcode

0x07: Device not available

0x08: Double fault

0x0E: Page fault

0x80: System call

Linux File Descriptors

0: stdin

1: stdout

2: stderr

Register Segments

EAX (Accumulator)

32-bit	EAX	
16-bit		AX
8-bit		AH / AL

ECX (Counter)

32-bit	ECX	
16-bit		CX
8-bit		CH / CL

EBX (Base)

32-bit	EBX	
16-bit		BX
8-bit		BH / BL

EDX (Data)

32-bit	EDX	
16-bit		DX
8-bit		DH / DL

ESI (Source Index)

32-bit	ESI	
16-bit		SI

ESP (Stack Pointer)

32-bit	ESP	
16-bit		SP

EDI (Destination Index)

32-bit	EDI	
16-bit		DI

EBP (Base Pointer)

32-bit	EBP	
16-bit		BP

Linux Process Signals

SIGHUP (1): Controlling terminal or parent process has stopped.

SIGINT (2): User interrupt from keyboard (Ctrl+C).

SIGQUIT (3): User interrupt from keyboard, with core memory dump (Ctrl+\)

SIGILL (4): Invalid/illegal machine instruction.

SIGTRAP (5): Pause program execution; used for debugging.

SIGABRT (6): Abort; process self-terminates due to irrecoverable error.

SIGBUS (7): Invalid memory access or hardware fault.

SIGFPE (8): Floating-point exception.

SIGKILL (9): Terminate process immediately; cannot be ignored or handled.

SIGUSR1 (10): User-defined signal. Does nothing by default.

SIGSEGV (11): Segmentation violation; illegal memory access.

SIGUSR2 (12): User-defined signal. Does nothing by default.

SIGPIPE (13): Tried to write to a closed pipe or socket.

SIGALRM (14): Alarm, typically the result of **alarm()**.

SIGTERM (15): Terminate process gracefully.

SIGSTKFLT (16): Stack overflow or stack-related fault; **ARM only**.

SIGCHLD (17): Child process has terminated or changed state.

SIGCONT (18): Resume a stopped or paused process.

SIGSTOP (19): Pause process; cannot be ignored or handled.

SIGTSTP (20): User-initiated pause; can be caught or handled. (Ctrl+Z)

SIGTTIN (21): Background process is trying to read from **stdin**.

SIGTTOU (22): Background process is trying to write to **stdout**.

SIGURG (23): Urgent data (OOB) received on socket.

SIGXCPU (24): Process is consuming too much CPU time, may be terminated.

SIGXFSZ (25): Writing file that exceeds the max allowed size RLIMIT_FSIZE

SIGVTALRM (26): ITIMER_VIRTUAL has elapsed.

SIGPROF (27): ITIMER_PROF has elapsed.

SIGWINCH (28): Controlling terminal's window size has changed.

SIGIO/SIGPOLL (29): Asynchronous I/O resource is ready (**legacy**).

SIGPWR (30): Power lost, running on battery (if one exists).

SIGSYS/SIGUNUSED (31): Invalid or non-existent system call.

x86 Instruction Set

Data

MOV dest, src - Copies **src** to **dest**

Arithmetic

ADD dest, src - Add **src** to **dest**

ADC dest, src – Add **src** & CF to **dest**

SUB dest, src - Take **src** from **dest**

SBB dest, src - Take **src** & CF from **dest**

INC dest – Increments **dest** by 1

DEC dest - Decrements **dest** by 1

MUL src - Unsigned multiply **src** by **EAX**

IMUL src - Signed multiply **src** by **EAX**

DIV src - Unsigned divide **src** by **EAX**

IDIV src - Signed divide **src** by **EAX**

Logic

AND dest, src - Bitwise AND

OR dest, src - Bitwise OR

XOR dest, src - Bitwise XOR

NOT dest - Bitwise NOT

Control Flow

JMP label - Unconditional jump

JE/JZ label - Jump if equal/zero

JNE/JNZ label - Jump if not equal/zero

CALL procedure - Call **procedure**/function

RET - Return from procedure

Stack Operations

PUSH src - Push **src** onto stack

POP dest - Pop **dest** from stack

Compare

CMP op1, op2 - Compare **op1** and **op2**

TEST op1, op2 - Bitwise AND comparison

Atomic

LOCK ADD dest, src – atomic **ADD**

Linux System Calls

Processes

fork() – create a child process

clone() – create a new execution space (process, thread, etc...)

exit() – terminate the current process and free resources

kill() – send a signal to a process (not necessarily SIGKILL)

Files

open() – open a *file* (any stream of data that can be read/written)

read() – read data from a file descriptor (an open file)

write() – write data to a file descriptor (an open file)

close() – close a file descriptor (an open file)

Network

socket() – Open one end of a communication channel (IPC or network)

bind() – Assign an address to an open socket (IPV4, IPV6 or IPC)

listen() – Mark socket as passive (accept incoming requests)

accept() – Block until a connection request is received

select() - Monitor file descriptors until one is ready for I/O

poll() -

connect() -

send() -

recv() -

getpid()

alarm()

sleep()

pipe()

shmget()

mmap()