

General Registers

EAX	Accumulator
EBX	Base
ECX	Counter
EDX	Data

General Registers: specific values are expected when calling the kernel.

Pointer-Registers

ESP	Stackpointer
EBP	Basepointer
EIP	Instructionpointer

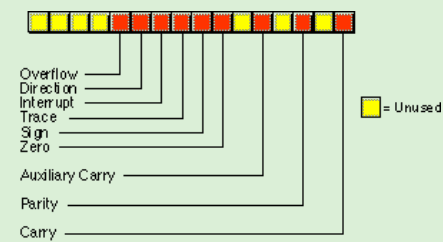
Index-Registers

ESI	Source Index
EDI	Destination Index

Segment- Registers

ECS	Code-Segment
EDS	Data-Segment
ESS	Stack-Segment
EES	Extra-Segment

Flags



NASM Basics

-f	filesystem
-g	debuginfos
-o	output

Compiling a Code

```
nasm -f elf32 -g -o filename.o
filename.nasm
ld -o filename filename.o
```

in 64bit Architecture use -f elf64

Syscall-Numbers Linux

EAX	Name(EBX, ECX, EDX)
1	exit(int)
2	fork(pointer)
3	read(uint, char*, int)
4	write(uint, char*, int)
5	open(char *, int, int)

[Linux Syscall Reference](#)

NASM Code-Sections

.text	Code
.data	initialized Data
.bss	uninitialized Data

Example

```
global _start
.data
    msg db "Hello World",0xa
    len equ $-msg
.text
_start:
```

Example (cont)

```
mov eax, 0x4
mov ebx, 0x1
mov ecx, msg
mov edx, len
int 0x80
exit:
mov eax, 0x1
mov ebx, 0x1
int 0x80
```

Misc

int <i>Nr</i>	call Interrup <i>Nr</i>
call <i>label</i>	jumps to <i>label</i>
ret	returns to call
nop	no operation
lea <i>dest, src</i>	load effective addr. to <i>dest</i>
int 0x80	calls the Kernel in Linux

Logical Operations

neg <i>op</i>	two-Complement
not <i>op</i>	invert each bit
and <i>dest, source</i>	$dest = dest \wedge source$
or <i>dest, source</i>	$dest = dest \vee source$
xor <i>dest, source</i>	$dest = dest \oplus source$



By **Gregor Lüdi** (Siniansung)
cheatography.com/siniansung/
www.ken.ch/%7Elueg

Published 13th January, 2015.
 Last updated 19th January, 2015.
 Page 1 of 2.

Sponsored by **CrosswordCheats.com**
 Learn to solve cryptic crosswords!
<http://crosswordcheats.com>

Control / Jumps (signed Int)

`cmp op1, op2` Compare *op1* with *op2*

`test op1, op2` bitwise comparison

`jmp dest` unconditional Jump

`je dest` Jump if equal

`jne dest` Jump if not equal

`jz dest` Jump if zero

`jnz dest` Jump if not zero

`jg dest` Jump if greater

`jge dest` Jump if greater or equal

`jl dest` Jump if less

`jle dest` Jump if less or equal

For unsigned Integer use `ja`, `jae` (above) or `jb`, `jbe` (below)

Mnemonics Intel

`mov dest, source` Moves Data

`add dest, value` Add *value* to *dest*

`sub dest, value` Subtract *value3* from *dest**

`inc dest` Increment *dest*

`dec dest` Decrement *dest*

`mul src` Multiply EAX and *src*

`imul dest, source` $dest = dest * source$

General Structure:

[label] mnemonic [operands]

[:comment]

Stack Operations

`push source` Insert Value onto the stack

`pop dest` Remove value from stack

Stack is a LIFO-Storage (Last In First Out)



By **Gregor Lüdi** (Siniansung)
cheatography.com/siniansung/
www.ken.ch/%7elueg

Published 13th January, 2015.
Last updated 19th January, 2015.
Page 2 of 2.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
<http://crosswordcheats.com>