F Assembler Cheat Sheet

General-purpose registers: `RAX`, `RBX`, `RCX`, `RDX`, `RSI`, `RDI`, `R8`, `R9`, `R10`, `R11`, `R12`, `R13`, `R14`, `R15`.
Special-purpose registers: `RSP` (stack pointer), `RBP` (base pointer), `RIP` (instruction pointer).

Operands: **r** = register, **m** = memory location, **im** = immediate.

Operand	Type	What it means
`$0`	**im**	decimal 0
`$0x10`	**im**	hexadecimal 10 (=16 decimal)
`lbl`	**m**	value stored at address of label `lbl`
`lbl+2`	**m**	value stored at two bytes after label `lbl`
`$lbl`	**im**	address of label `lbl`
`$(lbl+4)`	**im**	address of label `lbl` plus 4
`%rdx`	**r**	value stored in `RDX`
`(%rax)`	**m**	value at the address stored in `RAX`
`8(%rbp)`	**m**	value at eight bytes after the address stored in `RBP`
`-3(%rax)`	**m**	value at three bytes before the address stored in `RAX`

Instruction	**Operands**		**Operation**
< Mnemonic – Description >	*src*	*dest*	
`ADD` – Add	**r/m/im**	**r/m**	$dest \leftarrow dest + src$
`AND` – Bitwise logical AND	**r/m/im**	**r/m**	$dest \leftarrow AND(dest, src)$
`CALL` – Call procedure		**r/m/im**	push RIP, then $RIP \leftarrow dest$
`CMP` – Compare two operands	**r/m/im**	**r/m**	modify status flags similar to `SUB`
`CQO` – Sign-extend RAX to RDX before IDIV			$RDX : RAX \leftarrow sign - extendedRAX$
`DEC` – Decrement by 1		**r/m**	$dest \leftarrow dest - 1$
`IDIV` – Signed divide	**r/m**		signed divide $RDX : RAX$ by src $RAX \leftarrow quotient$, $RDX \leftarrow remainder$
`IMUL` – Signed multiply (2 op)	**r/m/im**	**r**	$dest \leftarrow dest * src$
`IMUL` – Signed multiply (1 op)	**r/m**		$RDX : RAX \leftarrow RAX * src$
`INC` – Increment by 1		**r/m**	$dest \leftarrow dest + 1$
`Jcc` – Jump if condition is met		**m/im**	conditionally $RIP \leftarrow dest$
`JMP` – Unconditional jump		**m/im**	$RIP \leftarrow dest$
`LEA` – Load effective address	**m**	**r**	$dest \leftarrow addressOf(src)$
`MOV` – Move	**r/m/im**	**r/m**	$dest \leftarrow src$
`NEG` – Two's Complement negation		**r/m**	$dest \leftarrow -dest$
`NOT` – One's Complement negation		**r/m**	$dest \leftarrow NOT(dest)$
`OR` – Bitwise logical OR	**r/m/im**	**r/m**	$dest \leftarrow OR(dest, src)$
`POP` – Pop value off the stack		**r/m**	$dest \leftarrow POP(stack)$
`PUSH` – Push value on the stack	**r/m/im**		$PUSH(stack, src)$
`RET` – Return from procedure			restore RIP by popping the stack
`SUB` – Subtract	**r/m/im**	**r/m**	$dest \leftarrow dest - src$
`SYSCALL` – System Call			invoke OS kernel

Operand size suffix: **b** = 1 byte, **w** = 2 bytes, **l** = 4 bytes, **q** = 8 bytes.
Use instruction mnemonic + suffix to get the instruction name. For example: `negq`, `movq`, `movl`.

Conditional jumps:

Instruction	Description
`JE`	Jump if equal
`JNE`	Jump if not equal
`JG`	Jump if greater than
`JGE`	Jump if greater than or equal
`JL`	Jump if less than
`JLE`	Jump if less than or equal

```
cmp op1, op2
jge lbl         # Jump to lbl if op2 >= op1.
```