Assembly Language Cheat Sheet

A quick reference for Assembly Language syntax, instructions, and concepts. Covers basic instructions, addressing modes, and common programming constructs.

Basic Instructions

Data Transfer Instructions

MOV — Move data from one location to another.

Example:

```
MOV destination, source
```

LEA — Load Effective Address. Loads the address of the source operand into the destination operand.

Example:

```
LEA destination, source
```

PUSH — Push data onto the stack.

Example:

```
PUSH source
```

POP — Pop data from the stack.

Example:

```
POP destination
```

XCHG — Exchange the contents of two operands.

Example:

```
XCHG operand1, operand2
```

Arithmetic Instructions

ADD — Add two operands.

Example:

```
ADD destination, source
```

SUB — Subtract two operands.

Example:

```
SUB destination, source
```

MUL — Multiply two operands.

Example:

```
MUL source
```

DIV — Divide two operands.

Example:

```
DIV source
```

INC — Increment operand by 1.

Example:

```
INC destination
```

DEC — Decrement operand by 1.

Example:

```
DEC destination
```

Logical Instructions

AND — Bitwise AND operation.

Example:

```
AND destination, source
```

OR — Bitwise OR operation.

Example:

```
OR destination, source
```

XOR — Bitwise XOR operation.

Example:

```
XOR destination, source
```

NOT — Bitwise NOT operation.

Example:

```
NOT destination
```

SHL / **SAL** — Shift Left / Shift Arithmetic Left.

Example:

```
SHL destination, count
```

SHR / **SAR** — Shift Right / Shift Arithmetic Right.

Example:

```
SHR destination, count
```

ROL — Rotate Left.

Example:

```
ROL destination, count
```

ROR — Rotate Right.

Example:

```
ROR destination, count
```

Control Flow

Comparison and Jump Instructions

`CMP`	Compare two operands (affects flags). **Example:** `CMP operand1, operand2`
`JE` / `JZ`	Jump if Equal / Jump if Zero. **Example:** `JE label`
`JNE` / `JNZ`	Jump if Not Equal / Jump if Not Zero. **Example:** `JNE label`
`JG` / `JNLE`	Jump if Greater / Jump if Not Less or Equal. **Example:** `JG label`
`JL` / `JNGE`	Jump if Less / Jump if Not Greater or Equal. **Example:** JL label
`JGE` / `JNL`	Jump if Greater or Equal / Jump if Not Less. **Example:** JGE label
`JLE` / `JNG`	Jump if Less or Equal / Jump if Not Greater. **Example:** JLE label
`JMP`	Unconditional jump. **Example:** `JMP label`

Looping

`LOOP`	Decrement `CX` and jump to label if `CX` != 0.
	Example:
	`MOV CX, 10 ; Set loop counter`
	`loop_start:`
	`; ... loop body ...`
	`LOOP loop_start`

Subroutines (Functions)

`CALL`	Call a subroutine. Pushes the return address onto the stack and jumps to the subroutine. **Example:** `CALL subroutine_name`
`RET`	Return from a subroutine. Pops the return address from the stack and jumps to it. **Example:** `RET`

Addressing Modes

Addressing Modes

Immediate Addressing: Operand is a constant value.

Example:

```
MOV AX, 10  ; Move the value 10 into register AX
```

Register Addressing: Operand is a register.

Example:

```
MOV AX, BX  ; Move the value in register BX to register AX
```

Direct Addressing: Operand is a memory address.

Example:

```
MOV AX, [1000]  ; Move the value at memory address 1000 into register AX
```

Indirect Addressing: Operand is a register that contains the memory address.

Example:

```
MOV AX, [BX]  ; Move the value at the memory address stored in register BX into register AX
```

Base-Plus-Index Addressing: Operand is a memory address calculated by adding a base register and an index register.

Example:

```
MOV AX, [BX + SI] ; Move the value at the memory address (BX + SI) into register AX
```

Base-Plus-Index with Displacement Addressing: Operand is a memory address calculated by adding a base register, an index register, and a displacement value.

Example:

```
MOV AX, [BX + SI + 10] ; Move value from memory address (BX + SI + 10) into AX
```

Data Directives

Data Definition Directives

`DB` Define Byte. Allocates one byte of storage.

Example:

```
my_byte DB 10 ; Defines a byte variable named my_byte with initial value 10
```

`DW` Define Word. Allocates two bytes of storage.

Example:

```
my_word DW 1000 ; Defines a word variable named my_word with initial value 1000
```

`DD` Define Doubleword. Allocates four bytes of storage.

Example:

```
my_dword DD 100000 ; Defines a doubleword variable named my_dword with initial value 100000
```

`DQ` Define Quadword. Allocates eight bytes of storage.

Example:

```
my_qword DQ 100000000 ; Defines a quadword variable named my_qword with initial value 100000000
```

`DT` Define Ten Bytes. Allocates ten bytes of storage.

Example:

```
my_tenbytes DT 1234567890 ; Defines a ten-byte variable
```

Other Directives

`EQU` Equate. Defines a constant value.

Example:

```
BUFFER_SIZE EQU 1024 ; Defines a constant BUFFER_SIZE with value 1024
```

`$` Represents the current address of the assembler.

Example:

```
JMP $ ; creates an infinite loop (jumps to the current instruction)
```