ARM Assembly Cheat Sheet

by Syshella via cheatography.com/128892/cs/26951/

Code Layout

```
.data
@variables
.text
.global main
@code
```

@ - Comments

Data Types

Type	Mnemonic	Bytes size
Word		4
Half word	h	2
Byte	b	1

Registers

#	Purpose	Description
R0 - R12	General Purpose	Stores temporary values, pointers...
R13	SP - Stack Pointer	Top of the stack. Allocate space to the stack by substracting the value in bytes that we want to allocate from the stack pointer.
R14	LR - Link Register	When a function call is made, LR gets pdated with a memory address referencing the next instruction where the function was initiated from
R15	PC - Program Counter	Automatically incremented by the size of the instruction executed

CPSR - Current Program Status Register

Flag	Meaning	Enable if result of the instruction yields a...
N	Negative	Negative number
Z	Zero	Zero value
C	Carry	Value that requires a 33rd bit to be fully represented
V	Overflow	Value that cannot be represented in 32 bit two's complement

Flexible operand

#123	Inmediate value
Rx	Register x
Rx, LSL n	Register x with logical shift left by n bits
Rx, LSR n	Register x with logical shift right by n bits

Syntax

MNEMONIC{S}{condition} {Rd}, Operand1, Operand2

Mnemonics

MNEMONIC	Description
{S}	An optional suffix. If S is specified, the condition flags are updated on the result of the operation
{condition}	Condition that is needed to be met in order for the instruction to be executed
{Rd}	Register destination for storing the result of the instruction
Operand1	First operand. Either a register or an inmediate value

Mnemonics (cont)

Operand2	Second (flexible) operand. Either an inmediate value (number) or a register with an optional shift

{} - Optional

Common Instructions

Instruction	Description
MOV	Move data
MVN	Move and negate
ADD	Addition
SUB	Substraction
MUL	Multiplication
LSL	Logical Shift Left
LSR	Logical Shift Right
ASR	Arithmetic Shift Right
CMP	Compare
CMN	Compare and negate
AND	Bitwise AND
ORR	Bitwise OR
EOR	Bitwise XOR
LDR	Load
STR	Store
LDM	Load Multiple
STM	Store Multiple
B	Branch
BL	Branch with Link
BX	Branch and eXchange
BLX	Branch with Link and eXchange
BIC	Bit Clear

Address modes

Offset

```
str r2, [r1, #2]
```

Store the value found in R2 to the memory address found in R1 plus 2. Base register unmodified.

Address modes (cont)

Pre-indexed

> `str r2, [r1, #4]!`
> Store the value found in R2 to the memory address found in R1 plus 4. Base register (R1) modified: R1= R1+4

Post-indexed

> `ldr r3, [r1], r2, LSL#2`
> Load the value at memory address found in R1 to the register R3. Then modify base register: R1 = R1+R2<<2

Syntax:

`STR Ra, [Rb, imm]`

`LDR Ra, [Rc, imm]`

If there is a !, its prefix address mode

`ldr r3, [r1, #4]!`

`ldr r3, [r1, r2]!`

`ldr r3, [r1, r2, LSL#2]!`

If the base register is in brackets by itself, it’s postfix address mode

`ldr r3, [r1], #4`

`ldr r3, [r1], r2`

`ldr r3, [r1], r2, LSL#2`

Anything else, offset address mode:

`ldr r3, [r1, #4]`

`ldr r3, [r1, r2]`

`ldr r3, [r1, r2, LSL#2]`

Conditionals

Mnemonic	Description	Flags
EQ	Equals	Z=1
NE	Non equals	Z=0
HI	Higher than (NS)	C=1 & Z=0
LS	Less than (NS)	C=0 | Z=1
GE	Greater or equals (WS)	N=V
LT	Less than (WS)	N!=V
GT	Greater than (WS)	Z=0 & N=V
LE	Less or equals than (WS)	Z=1 | N!=V
(empty)	Always (non conditional)	

NS - No sign
WS - With sign

Most of intructions can be executed using conditionals. Ie:
`movle r2, r1