ARM Assembly Operations

Simplest Complete Program (compile with `gcc -o filename filename.s`; run with `./filename`)

```
.global main
main:
    mov r7, #1          @exit system call
    svc #0
```

Basic operations

Argument `dr` is the register in which to store the result. Operands `or` must be a register (e.g. `r1`). Operands `oi` can be a register or an immediate (e.g. `#5`). The argument `#0` for `svc` must be this value.

Add	`add dr, or, oi`
Subtract (`or` − `oi`)	`sub dr, or, oi`
Reverse subtract (`oi` − `or`)	`rsb dr, or, oi`
Multiply (`dr` and `or1` cannot be the same)	`mul dr, or1, or2`
Divide signed numbers[1](`or1` / `or2`)	`sdiv dr, or1, or2`
Logical shift left (`oi` must be immediate)	`lsl dr, or, oi`
Copy (from `oi` to `dr`)	`mov dr, oi`
Compare `or` to `oi` and set comparison flags	`cmp or, oi`
Branch to `label`	`b` *`address`*
Branch and link	`bl` *`address`*
System call (see table below)	`svc #0`

`svc #0` is controlled by the contents of register `r7`:

1	Exit program
3	Read string (`r2` bytes long) and store using address in `r1`. `r0` must be #0 (standard input)
4	Print string (`r2` bytes long) whose address is stored in `r1`. `r0` must be #1 (standard output)

Conditional Suffixes

All instructions can be used conditionally (based on the last call to `cmp`) by adding one of these suffixes.

If flags are set to "equal"	`eq`	If flags are set to "not equal"	`ne`
If flags are set to "greater than or equal"	`ge`	If flags are set to "less than or equal"	`le`
If flags are set to "greater than"	`gt`	If flags are set to "less than"	`lt`

Memory instructions

Switch to the text segment	`.text`
Switch to the data segment	`.data`
Store `str` as a null-terminated string	`.asciz "str"`
Reserve `oi` bytes of space (`oi` must be immediate)	`.space oi`
Create word (`or` can be a string)	`.word or`
Load word from *`address`*	`ldr dr,` *`address`*
Load address of *`labelText`*	`ldr dr, =#`*`labelText`*
Store word at *`address`*	`str or,` *`address`*
Load byte from *`address`*	`ldrb dr,` *`address`*
Store byte at *`address`*	`strb or,` *`address`*
Push register values to the stack	`push {`*`reglist`*`}`
Pop register values from the stack	`pop {`*`reglist`*`}`

`address` can be `[or]` or `[or, oi]`, with the values being added together when `oi` is provided
`reglist` is a comma-separated list of registers and ranges of registers (e.g. `r1, r5-r7`)
one more line

[1]Requires the compile flag `-mcpu=cortex-a7` for `gcc`; see `https://forums.raspberrypi.com/viewtopic.php?t=320122`