ARM Assembly Language Reference

|  |  |
| --- | --- |
| **Comments** | Single line comments:  @ comment  // comment  Multi-line comments:  /\* comment \*/ |
| **Load constant into register** | 8 bit immediate:  mov r0, #10  16 bit immediate:  movw r0, #0x1010  32 bit immediate (using two instructions):  movw r0, #0x1010  movt r0, #0x2222  /\* result is 0x22221010 \*/  32 bit immediate (using a pseudoinstruction):  ldr r0, =0x22221010 |
| **Copy one register into another** | Set r4 to contain the contents of r2  mov r4, r2 |
| **Load from memory** | Load r0 with the address that’s stored in r1:  ldr r0, [r1]  Load r0 with an address that’s 4 bytes plus r1:  ldr r0, [r1,#4] |
| **Store to memory** | Store r0 into the address that’s stored in r1:  str r0, [r1]  Store r0 into the address that’s 8 bytes plus r1:  str r0, [r1,#8] |
| **Add** | Add r0 += 10  add r0, #10  Add r0 = r1 + r2  add r0, r1, r2 |
| **Subtract** | Subtract r0 -= 1  sub r0, #1  Subtract r0 = r1 – r2  sub r0, r1, r2 |
| **Compare** | Update condition flags by comparing r3 with 0  cmp r3, #0 |
| **Conditional branch** | mov r0, #10 /\* r0 = 10 \*/  loop:  sub r0, #1 /\* r0 -= 1 \*/  cmp r0, #0 /\* is r0 equal to zero? \*/  bne loop /\* branch if not equal \*/ |

|  |  |
| --- | --- |
| **Standard function prologue and epilogue** | .text  .align 2  main:  push {r4-r11, lr}  /\* ... \*/  pop {r4-r11, pc} |
| **Calling a function** | // place arg1 into r0  // place arg2 into r1  // ...  bl func  // return value is in r0 |
| **Allocating memory on the stack** | sub sp, #8 /\* number of bytes needed \*/    // these two integers are at [sp] and [sp,#4]  add sp, #8 /\* clean up the stack \*/ |
| **Allocating memory statically** | .data  .align 2  variable\_name:  .skip 12 /\* number of bytes \*/ |
| **Declaring constants** | Integer constant:  .text  .align 2  const1:  .word 0x10102323  ASCII string:  .text  .align 2  str1:  .ascii "String\0" /\* notice trailing NULL \*/ |