|  |  |  |
| --- | --- | --- |
| ARM Reference  This page contains a reference for the ARMv6 instruction set, register set, and also the GNU Assembler program syntax.   |  | | --- | | **Contents**   * [1 Tutorial ARM Instructions](http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/armv6.html#tutorial) |   1 Tutorial ARM Instructions  The following is a list of all the instruction boxes in the courses in order.  **ldr reg,=val** puts the number **val** into the register named **reg**.  **mov reg,#val** puts the number **val** into the register named **reg**.  **lsl reg,#val** shifts the binary representation of the number in **reg** by **val** places to the left.  **str reg,[dest,#val]** stores the number in **reg** at the address given by **dest** + **val**.  **name:** labels the next line **name**.  **b label** causes the next line to be executed to be **label**.  **sub reg,#val** subtracts the number **val** from the value in **reg**.  **cmp reg,#val** compares the value in **reg** with the number **val**.  Suffix **ne** causes the command to be executed only if the last comparison determined that the numbers were not equal.  **.globl lbl** makes the label **lbl** accessible from other files.  **mov reg1,reg2** copies the value in **reg2** into **reg1**.  Suffix **ls** causes the command to be executed only if the last comparison determined that the first number was less than or the same as the second. Unsigned.  Suffix **hi** causes the command to be executed only if the last comparison determined that the first number was higher than the second. Unsigned.  **push {reg1,reg2,...}** copies the registers in the list **reg1,reg2,...** onto the top of the stack. Only general purpose registers and lr can be pushed.  **bl lbl** sets **lr** to the address of the next instruction and then branches to the label **lbl**.  **add reg,#val** adds the number **val** to the contents of the register **reg**.  Argument shift **reg,lsl #val** shifts the binary representation of the number in **reg** left by **val** before using it in the operation before.  **lsl reg,amt** shifts the binary representation of the number in **reg** left by the number in **amt**.  **str reg,[dst]** is the same as **str reg,[dst,#0]**.  **pop {reg1,reg2,...}** copies the values from the top of the stack into the register list **reg1,reg2,...**. Only general purpose registers and pc can be popped.  **alias .req reg** sets **alias** to mean the register **reg**.  **.unreq alias** removes the alias **alias**.  **lsr dst,src,#val** shifts the binary representation of the number in **src** right by **val**, but stores the result in **dst**.  **and reg,#val** computes the Boolean and function of the number in **reg** with **val**.  **teq reg,#val** checks if the number in **reg** is equal to **val**.  **ldrd regLow,regHigh,[src,#val]** loads 8 bytes from the address given by the number in **src** plus **val** into **regLow** and **regHigh**.  **.align num** ensures the address of the next line is a multiple of 2**num**.  **.int val** outputs the number **val**.  **tst reg,#val** computes **and reg,#val** and compares the result with 0.  **mla dst,reg1,reg2,reg3** multiplies the values from **reg1** and **reg2**, adds the value from **reg3** and places the least significant 32 bits of the result in **dst**.  **strh reg,[dest]** stores the low half word number in **reg** at the address given by **dest**. |  |