

**explain differences: }Pre-index**

LDR r1, [r0, #4]

}Post-index  
  LDR r1, [r0], #4

}Pre-index with Update  
  LDR r1, [r0, #4]!

Here's a concise explanation of the differences between the pre-index, post-index, and pre-index with update ARM load instructions from the image:[[1]](#fn1)[[2]](#fn2)[[3]](#fn3)

**Pre-index: LDR r1, [r0, #4]**

* Calculates the effective address by adding the offset (here, #4) to the base register (r0) before the load.
* Loads data from the resulting address into r1.
* The base register (r0) is **not** updated.
* Example:
  + If r0 = 0x100, instruction accesses memory at 0x104, but r0 remains 0x100 after execution.[[2]](#fn2)[[3]](#fn3)

**Post-index: LDR r1, [r0], #4**

* Loads data from the address currently in r0 into r1.
* After the load, updates the base register (r0) by adding the offset (#4).
* Example:
  + If r0 = 0x100, instruction accesses memory at 0x100, then sets r0 to 0x104 after.[[3]](#fn3)[[2]](#fn2)

**Pre-index with Update: LDR r1, [r0, #4]!**

* First, adds the offset (#4) to the base register (r0), then loads from this updated address.
* Updates the base register: r0 is set to r0 + 4 after execution.
* Example:
  + If r0 = 0x100, instruction accesses memory at 0x104 and also sets r0 to 0x104 afterward.[[1]](#fn1)[[2]](#fn2)[[3]](#fn3)

**Summary Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Mode | Address used for Load | Base register update | Example (r0=0x100) |
| Pre-index | r0 + offset (0x104) [[2]](#fn2) | No | r1 = data[0x104]; r0 = 0x100 |
| Post-index | r0 (0x100) [[3]](#fn3) | Yes, after load | r1 = data[0x100]; r0 = 0x104 |
| Pre-index w/ Update | r0 + offset (0x104) [[1]](#fn1) | Yes, before load | r1 = data[0x104]; r0 = 0x104 |

These modes help optimize common data access patterns (table walks, stack manipulation, pointer iteration) in ARM assembly.[[2]](#fn2)[[3]](#fn3)[[1]](#fn1)

⁂

1. <https://developer.arm.com/documentation/den0042/latest/Unified-Assembly-Language-Instructions/Memory-instructions/Addressing-modes>

1. <https://stackoverflow.com/questions/27998260/ldr-in-assembly-language-diffrent-forms>

1. <https://devblogs.microsoft.com/oldnewthing/20210602-00/?p=105271>

1. <https://azeria-labs.com/memory-instructions-load-and-store-part-4/>

1. <http://www-mdp.eng.cam.ac.uk/web/library/enginfo/mdp_micro/lecture4/lecture4-2-4.html>

1. <https://www.csie.ntu.edu.tw/~cyy/courses/assembly/12fall/lectures/handouts/lec09_ARMisa.pdf>

1. [https://www.eng.auburn.edu/~nelson/courses/elec2220/slides/ARM prog model 2 addressing.pdf](https://www.eng.auburn.edu/~nelson/courses/elec2220/slides/ARM%20prog%20model%202%20addressing.pdf)

1. <https://www.labs.cs.uregina.ca/301/ARM-addressing/lecture.html>

1. <https://developer.arm.com/documentation/100235/0004/the-cortex-m33-instruction-set/memory-access-instructions/ldr-and-str--immediate-offset?lang=en>

1. <https://duetorun.com/blog/20230613/arm-addressing-modes/>

1. <https://www.phlamethrower.co.uk/riscos/armforbeginners/3/index.htm>

1. image.jpg

1. <https://developer.arm.com/documentation/102374/latest/Loads-and-stores---addressing>

1. <https://www.youtube.com/watch?v=zgkxPdPkxa8>

1. <https://stackoverflow.com/questions/66107147/instruction-writeback-in-arm>

1. [https://ee315.cankaya.edu.tr/uploads/files/2024/EE 315 LAB2.pdf](https://ee315.cankaya.edu.tr/uploads/files/2024/EE%20315%20LAB2.pdf)

1. <https://stackoverflow.com/questions/75996910/arm-cortex-m3-reference-manual-mentioning-ldr-rx-any-as-a-valid-instruction>