The LDI Instruction

The High Level
The **Load Indirect** instruction is a type of **Data Movement Instruction** that, given a label such as ADDR_1 loads MEM[ADDR_1] into a register.

To understand the content of this tutorial, you should know what a **PCOffset9** is, as well as how the **LD** instruction works.

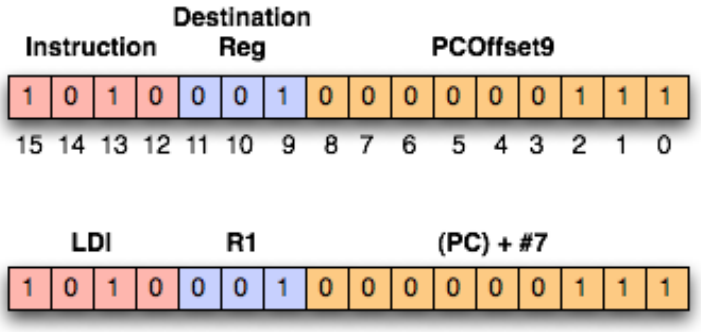

Figure 1: The LDI Instruction (details)

The Breakdown:

- Bits [15-12] specify what instruction to execute (1010 is LDI)
- Bits [11-9] specify which register to store the indirect value in
- Bits [8-0] specify a 9-bit PCOffset, which is the number of memory locations (i.e. "spaces") between the PC and the indirect value you want

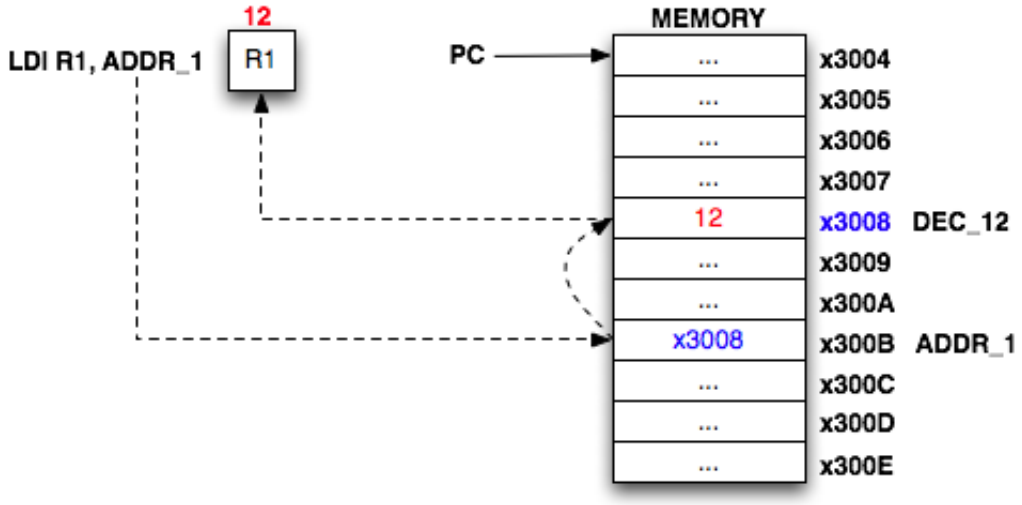

Figure 2: The LDI Instruction – Visual Execution

The Examples!

```
.orig x3000             ; program begins here
   ;---------------
   ; Instructions
   ;---------------
   LDI R1, ADDR_1       ; R1 <-- MEM[ADDR_1] == MEM[x4000] == 12
   HALT

   ;---------
   ; Data
   ;---------
   ADDR_1   .FILL  x4000

; Hard-code the value #12 at memory location x4000
.orig x4000
   .FILL  #12

.end
```

Pitfalls... (aka: Erroneous code makes baby fishies cry)
The example below is erroneous. Please do NOT try to code this way!

```
LDI R1, x4000  ; (ERROR: You must use a label,not a literal memory address)
LDI R1, LabelThatIsReallyFarAway ; (ERROR: Overflows 9-bit PCOffset9 field)
```

The first example pitfall code above is incorrect because you have to use a **label** whenever you use the LDI instruction. You cannot give the instruction an address. It's just not built that way.

The second example pitfall code above is incorrect and classically causes problems for **many** students. Since this instruction uses a 9-bit **PC Offset**, which means the indirect value you want to load must be no more than [Range of 9 bits] memory locations away from the PC (Note: The range of a Two's Compliment 9-bit field is ±[Range of 8 bits] == [-256, 255].