The STI Instruction

The High Level

The **Store Indirect** instruction is a type of **Data Movement Instruction** that, given a label such as ADDR_1 puts the value from a specified register into MEM[ADDR_1].

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

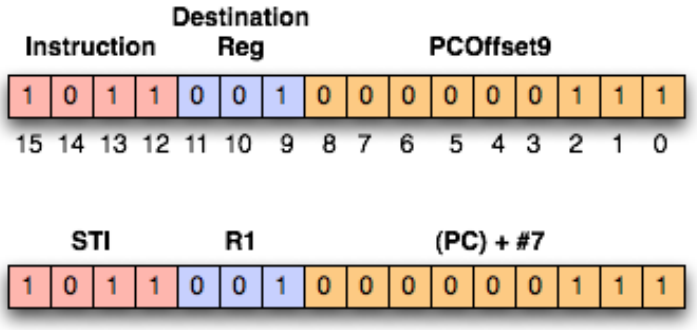

Figure 1: The STI Instruction (details)

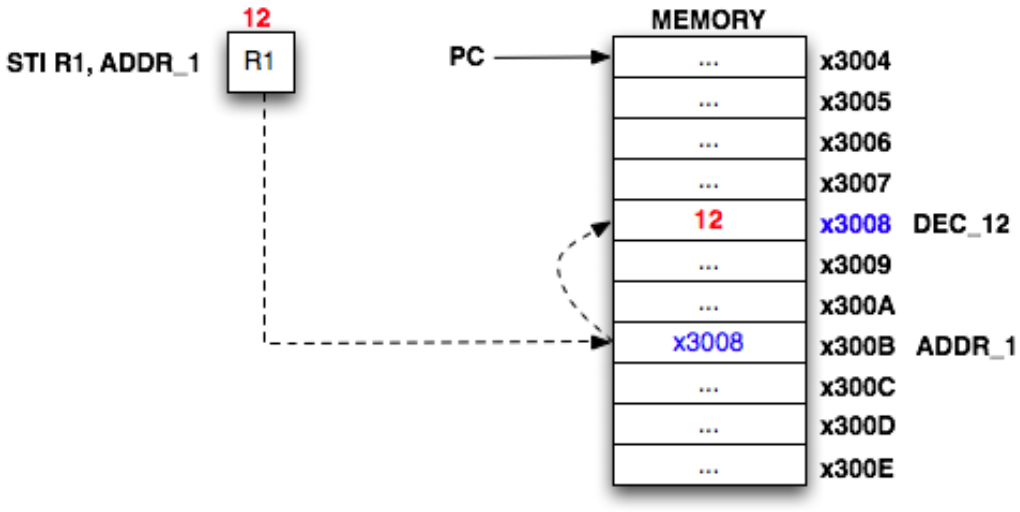

Figure 2: The STI Instruction – Visual Execution

The Breakdown:

- Bits [15-12] specify what instruction to execute (1011 is STI)
- Bits [11-9] specify which register's value to indirectly store
- Bits [8-0] specify a 9-bit PCOffset, which is the number of memory locations (i.e. "spaces") between the PC and the indirect location to store your value at

The Examples!

```
.orig x3000            ; program begins here
   ;---------------
   ; Instructions
   ;---------------
   AND R1, R1, #0      ; R1 <-- 0
   ADD R1, R1, #12     ; R1 <-- R1 + #12
   STI R1, ADDR_1      ; MEM[ADDR_1] <-- R1   (i.e. MEM[x4000] <-- R1)
   HALT

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

.end
```

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

```
   STI R1, x4000  ; (ERROR: You must use a label,not a literal memory address)
   STI 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 STI 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 location that you want to store to 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].