The ST Instruction

The High Level
The **Store Direct** instruction is a type of **Data Movement Instruction** that puts the value of a register directly into a memory location denoted by a **label**.

To do so, it uses a **PC Offset**, which is a count of how many memory locations ahead of the **Program Counter** (aka: the PC) the place you want to store the value is. In order words, the **PCOffset9** is like saying, "how many spaces ahead of the PC do I have to go to get to the place I want to store this value?"

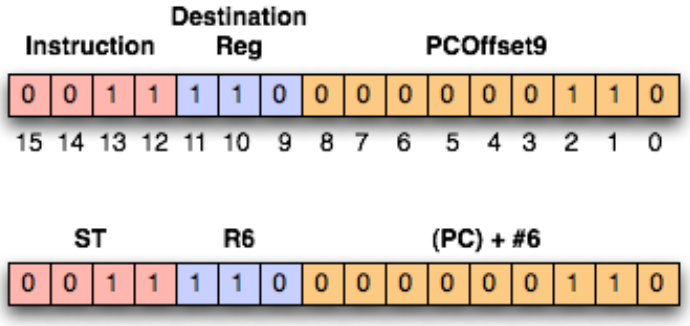

Figure 1: The ST Instruction (details)

The Breakdown:

- Bits [15-12] specify what instruction to execute (0011 is ST)
- Bits [11-9] specify which register's value to put into memory
- Bits [8-0] specify a 9-bit PCOffset, which is the number of memory locations (i.e. "spaces") between the PC and the place you want to store the value at

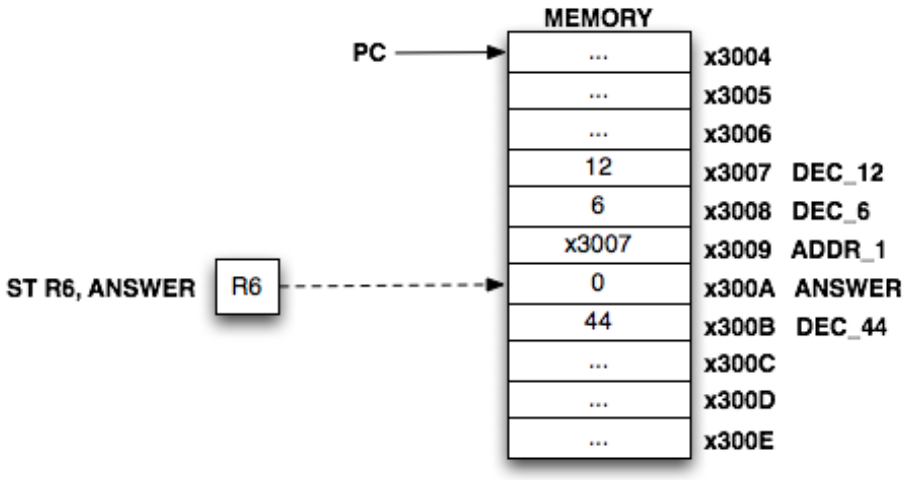

Figure 2: The ST Instruction – Visual Execution

The Examples!

```
.orig x3000            ; program begins here
   ;---------------
   ; Instructions
   ;---------------
   AND R1, R1, #0      ; Set R1 to 0       (Protip: This is *EXTREMELY* useful)
   ADD R1, R1, #15     ; R1 <-- R1 + #15
   ADD R1, R1, #15     ; R1 <-- R1 + #15
   ADD R1, R1, #12     ; R1 <-- R1 + #12

   ST R1, THE_ANSWER   ; MEM[THE_ANSWER] <-- R1

   HALT                ; Stop execution of program

   ;---------
   ; Data
   ;---------
   THE_ANSWER   .BLKW   #1
.end                   ; tell assembler not to look for any more code
```

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

```
ST R1, x4000  ; (ERROR: You must use a label,not a literal memory address)
ST R2, 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 ST 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 place you want to store a register's value 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].

Further Illustrated erroneous example:
If you have a program that begins at x3000, is 500 lines of code long, has as one of its first instruction a ST instruction, and has the **Data Area** at the end of the **Instruction Area** (as it should), then when the program tries to issue the ST instruction, it will find that it can't "reach" the desired storage location because it is "too far away" (i.e. cannot be reached with Two's Compliment 9-bits of range)

```
;-------------------------------------------------------------------
; ERRONEOUS CODE EXAMPLE:
; Example of a very common error that students run into.
; This program will result in an "Overflows 9-bit PCOffset" error
;-------------------------------------------------------------------
.orig x3000
   ;---------------
   ; Instructions
   ;---------------
   AND R2, R2, #0           ; R2 <-- 0
   ADD R2, R2, #10          ; R2 <-- R2 + #10
   ST R2, IMPORTANT_VALUE   ; MEM[IMPORTANT_VALUE] <-- R2

   ...
   ... [256 or more lines of code, thus pushing the Data Area
   ...  beyond the reach of the PCOffset9 that the LD instruction
   ...  needs to use]

   HALT
   ;-------
   ; Data
   ;-------
   IMPORTANT_VALUE     .BLKW     #1
.end
```