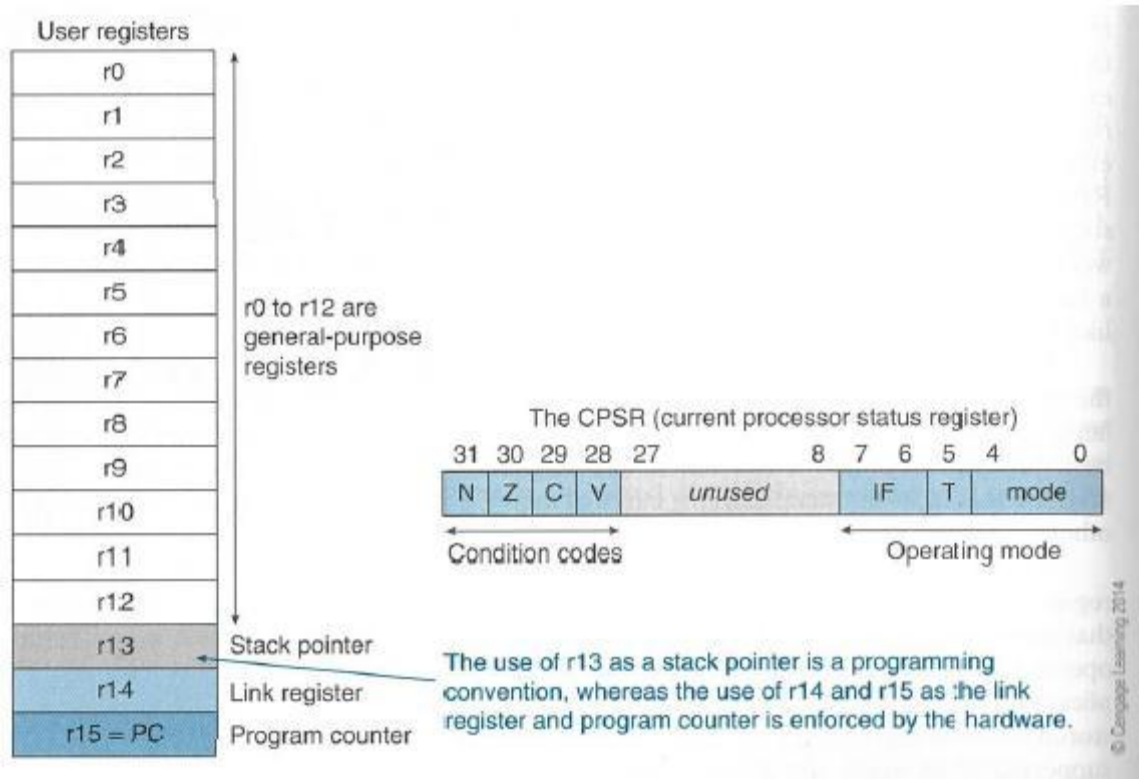
ARM's Flow Control Instructions

ARM's Flow Control Instructions modify the default sequential execution. They control the operation of the processor and sequencing of instructions.

Review of ARM Register Set

As mentioned in the previous lab, ARM has 16 programmer-visible registers and a *Current Program Status Register*, CPSR.
Here is a picture to show the **ARM register set**.




```
R0 to R12 are the general-purpose registers.
R13 is reserved for the programmer to use it as the stack pointer.
R14 is the link register which stores a subroutine return address.
R15 contains the program counter and is accessible by the programmer.

Conditonion code flags in CPSR:
N - Negative or less than flag
Z - Zero flag
C - Carry or bowrrow or extendedflag
V - Overflow flag
The least-significant 8-bit of the CPSR are the control bits of the system.
The other bits are reserved.
```

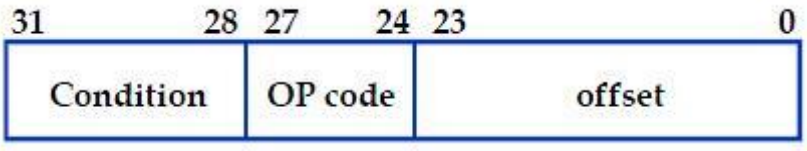
Setting Condition Code Flags

Some instructions, such as Compare, given by **CMP Rn, Rm** which performs the operation [Rn]-[Rm] have the sole purpose of setting the condition code flags based on the result of the subtraction operation.

The arithmetic and logic instructions affect the condition code flags only if explicitly specified to do so by a bit in the OP-code field. This is indicated by appending the suffix S to the OP-code.
For example, the instruction **ADDS R0, R1, R2** sets the condition code flags. But **ADD R0, R1, R2** does not.

The Encoding Format for Branch Instructions

Conditional branch instructions contain a signed 24-bit offset that is added to the updated contents of the Program Counter to generate the branch target address. Here is the encoding format for the branch instructions:



Offset is a signed 24-bit number. It is shifted left two-bit positions (all branch targets are aligned word addresses), signed extended to 32 bits, and added to the updated PC to generate the branch target address. The updated PC points to the instruction that is two words (8 bytes) forward from the branch instruction.

ARM instructions are conditionally executed depending on a condition specified in the instruction. The instruction is executed only if the current state of the processor condition code flag satisfies the condition specified in bits b31-b28 of the instruction. Thus the instructions whose condition does not meet the processor condition code flag are not executed. One of the conditions is used to indicate that the instruction is always executed.

Branch and Control Instructions

Branch instructions are very useful for selection control and looping control.

Here is a list of the ARM processor's Branch and Control instructions.

```
-----------------------------------------------------------------------
       B loopA         ; Branch to label loopA unconditioally
-----------------------------------------------------------------------
       B.W target      ; Branch to target within 16MB range
-----------------------------------------------------------------------
       BEQ target      ; Conditionally branch to target, when Z = 1
-----------------------------------------------------------------------
       BEQ.W target    ; Conditionally branch to target within 1MB when Z = 1
-----------------------------------------------------------------------
       BNE AAA         ; branch to AAA when Z = 0
-----------------------------------------------------------------------
       BMI BBB         ; branch to BBB when N = 1
-----------------------------------------------------------------------
       BPL  CCC        ; branch to CCC when N = 0
-----------------------------------------------------------------------
       BLT  labelAA    ; Conditionally branch to label labelAA,
                       ; N set and V clear  or  N clear and V set
                       ; i.e.  N != V
-----------------------------------------------------------------------
       BLE labelA      ; Conditionally branch to label labelA,
                       ; when less than or equal, Z set or N set and V clear
                       ; or N clear and V set
```

```
                        ; i.e.  Z = 1 or N != V
-------------------------------------------------------------------------
        BGT  labelAA    ; Conditionally branch to label labelAA,
                        ; Z clear and either N set and V set
                        ;             or  N clear and V clear
                        ; i.e.   Z = 0 and N = V
-------------------------------------------------------------------------
        BGE labelA      ; Conditionally branch to label labelA,
                        ; when Greater than or equal to zero,
                        ; Z set or N set and V clear
                        ; or N clear and V set
                        ; i.e.  Z = 1 or N !=V
-------------------------------------------------------------------------
        BL funC         ; Branch with link (Call) to function funC,
                        ; return address stored in LR, the register R14
-------------------------------------------------------------------------
        BX LR           ; Return from function call
-------------------------------------------------------------------------
        BXNE R0         ; Conditionally branch to address stored in R0
-------------------------------------------------------------------------
        BLX R0          ; Branch with link and exchange (Call)
                        ; to a address stored in R0.
-------------------------------------------------------------------------
```

Examples of Compare Instructions

```
        Mnemonic                  Meaning
-------------------------------------------------------------------------
        CBZ R5, target          ; Forward branch if R5 is zero
-------------------------------------------------------------------------
        CBNZ R0, target         ; Forward branch if R0 is not zero
-------------------------------------------------------------------------
        CMP R2, R9              ; update the N, Z, C and V flags
-------------------------------------------------------------------------
        CMN R0, #6400           ; update the N, Z, C and V flags
-------------------------------------------------------------------------
        CMPGT SP, R7, LSL #2    ; update the N, Z, C and V flags
-------------------------------------------------------------------------
```

Here are three links for your references.

1. **ARM branch instructions** from ARM Information Center.

2. **Cortex-M3 Devices Generic User Guide** Section 3.9.

3. **A link to ARM Instruction Set.**

An Example of Using Branch Instructions

```
;The semicolon is used to lead an inline documentation
;
;When you write your program, you could have your info at the top document block
;For Example: Your Name, Student Number, what the program is for, and what it does etc.
;
;       This program will count the length of a string.
;
```

```
;;; Directives
          PRESERVE8
          THUMB

; Vector Table Mapped to Address 0 at Reset
; Linker requires __Vectors to be exported

          AREA    RESET, DATA, READONLY
          EXPORT  __Vectors

__Vectors
          DCD  0x20001000     ; stack pointer value when stack is empty
          DCD  Reset_Handler  ; reset vector

          ALIGN

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Byte array/character string
; DCB type declares that memory will be reserved for consecutive bytes
; You can list comma separated byte values, or use "quoted" characters.
; The ,0 at the end null terminates the character string. You could also use "\0".
; The zero value of the null allows you to tell when the string ends.
;*****************************************************************************
string1
        DCB     "Hello world!",0

; The program
; Linker requires Reset_Handler

        AREA    MYCODE, CODE, READONLY

        ENTRY
        EXPORT Reset_Handler


Reset_Handler

;;;;;;;;;;User Code Start from the next line;;;;;;;;;;;;


        LDR     R0, = string1   ; Load the address of string1 into the register R0

        MOV     R1, #0          ; Initialize the counter counting the length of string1

loopCount

        LDRB    R2, [R0]        ; Load the character from the address R0 contains

        CBZ     R2, countDone   ; If it is zero...remember null terminated...
                                ; You are done with the string. The length is in R1.

        ADD     R0, #1;         ; Otherwise, increment index to the next character

        ADD     R1, #1;         ; increment the counter for length

        B       loopCount



countDone

STOP
        B  STOP
```

```
        END                     ; End of the program
```

Another Example

```
;The semicolon is used to lead an inline documentation
;When you write your program, you could have your info at the top document block
;For Example:  Your Name, Student Number, what the program is for, and what it does etc.
;
;       See if you can figure out what this program does
;

;;; Directives
          PRESERVE8
          THUMB

; Vector Table Mapped to Address 0 at Reset
; Linker requires __Vectors to be exported

          AREA    RESET, DATA, READONLY
          EXPORT  __Vectors

__Vectors
          DCD  0x20001000     ; stack pointer value when stack is empty
          DCD  Reset_Handler  ; reset vector

          ALIGN

;Your Data section
        ;AREA DATA

SUM     DCD 0
SUMP    DCD SUM


N       DCD 5


; The program
; Linker requires Reset_Handler

          AREA    MYCODE, CODE, READONLY

          ENTRY
          EXPORT Reset_Handler


Reset_Handler

;;;;;;;;;;User Code Start from the next line;;;;;;;;;;;


        LDR R1, N               ;Load count into R1

        MOV R0, #0              ;Clear accumulator R0

LOOP
        ADD R0, R0, R1          ;Add number into R0
        SUBS R1, R1, #1         ;Decrement loop counter R1
        BGT LOOP                ;Branch back if not done

        LDR R3, SUMP            ;Load address of SUM to R3
```

```
        STR R0, [R3]            ;Store SUM

STOP
        B  STOP

        END
```

Lab Assignment

This page last modified: Monday, 25-May-2015 16:22:51 CST	Accessed 036503 times.