# **ATMEL Studio 7 (DataAndLabels.asm)**

;PROJECT :DataAndLabels

;PURPOSE :Investigation of use of data and array access

;AUTHOR :C. D'Arcy

;DATE :2020 04 20

;DEVICE :Dolgin Development Platform

;MCU :ATtiny84

;COURSE :ICS4U

;STATUS :Working

.def util = r16 ;use aliases for GP Registers

; DATA Segment declarations

.dseg ;locate for Data Segment (SRAM) requirements (default start at 0x0060)

.org SRAM\_START

numbers: .BYTE 8 ;reserve data storage (the label is the symbol)

; CODE Segment

.cseg ;locate for Code Segment (FLASH)

; \*\*\*\*\* INTERRUPT VECTOR TABLE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.org 0x0000 ;start of Interrupt Vector Table (IVT) aka. Jump Table

rjmp reset ;lowest interrupt address == highest priority!

; \*\*\*\*\* START OF CODE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.org 0x0011 ;above the IVT

varStart: ;label (address) for start of data sequence stored in FLASH

.DB 'R','G','R','G'

varEnd: ;label (address) for end of this data sequence (useful)

arrStart: ;label (address) to mark the start of an array with initialization

.DB 2,0 ;reserves 2 bytes (not sure if second parameter used for fill)

arrEnd: ;(past) end of array

str1Start: ;start address of str1 array

.DB "RSGC" ;

str1End: ;(past) end of str1 array

str2Start: ;start address of str2 array

.DB "ACES.",0 ;append a 0 as necessary to to maintain even byte boundaries

str2End: ;(past) end of str2 array

str3Start: ;start address of st3 array

.DB "AVR Assembly" ;

str3End: ;(past) end of str3 array

.org 0x0100 ;a good place to start

reset: ;PC jumps to here (start of code) on reset interrupt...

ser util ;prepare to set ALL port bits for output

out DDRA,util ;set ddr pins for output

ldi XL,low(str3Start<<1) ;set 16-bit X register to point to the start of array

ldi XH,high(str3Start<<1) ;this requires 2 statements for the low and high bytes

ldi YL,low(str3End<<1) ;set 16-bit Y register to point to the start of array

ldi YH,high(str3End<<1) ;this requires 2 statements for the low and high bytes

again:

movw Z,X ;use 2-byte move to position Z as array index pointer

next:

;use post-increment mode to acquire byte from current index and advance the pointer

lpm util,Z+

out PORTA,util ;place the byte on the output port

rcall delay1s ;admire

cp ZL,YL ;are we at the end of the array (Z==Y?)

brne next ;if not, continue to next byte in the array

rjmp again ;if so, start over, do continuously

ret ;return

delay1s:

; Assembly code auto-generated

; by utility from Bret Mulvey

; Delay 8 000 000 cycles

; 1s at 8.0 MHz

ldi r18, 41

ldi r19, 150

ldi r20, 128

L1: dec r20

brne L1

dec r19

brne L1

dec r18

brne L1

ret

# Arduino IDE (DataAndLabels.S)

**// PROJECT  :DataAndLabels**

**// PURPOSE  :Investigation of the use of data declarations and array access with X,Y,and Z Registers**

**// COURSE   :ICS4U**

**// AUTHOR   :C. D'Arcy**

**// DATE     :2020 04 20**

**// MCU      :84**

**// STATUS   :Working**

// REFERENCE:<http://darcy.rsgc.on.ca/ACES/TEI4M/Assembly/images/GPRegisters.jpg>

#include  <avr/io.h>            //required to reference ports by name (-0x20 offset required)

#define util r16                //improve readability with aliases

.equ    OFFSET, 0x20            ;avr-as requires offset address for PORTs

.equ    PORT,PORTA-OFFSET       ;0x3B – 0x20 = 0x1B

.equ    DDR, DDRA-OFFSET        ;0x3A – 0x20 = 0x1A

**.section .data**     ;SRAM segment directive (use of .section is optional)

.org RAMSTART      ;optional (0x060 is first available address) See:iotn84.h

numbers:           ;a label is the identifier (name of variable)

.byte 8            ;reserves # of byte(s) of SRAM storage

**.section .text**     ;FLASH segment directive

.org  0x0000       ;set the Program Counter (PC) to the start of program memory (IVT)

  rjmp     main   ;jump to the beginning of executable code

.org 0x11          ;set the Program Counter (PC) beyond the start of the 84's IVT

varStart:          ;label (address) for start of data sequence stored in FLASH

.byte 'R','G','R','G'

varEnd:            ;label (address) for end of this data sequence (useful)

arrStart:          ;label (address) to mark the start of an array with initialization

.space 2,66        ;reserves 2 bytes and (optionally) initializes each to 66

arrEnd:            ;(past) end of array

str1Start:         ;start address of str1 array

.ascii  "RSGC"     ;

str1End:           ;(past) end of str1 array

str2Start:         ;start address of str2 array

.asciz  "ACES."    ;appends 0 at end

str2End:           ;(past) end of str2 array

str3Start:         ;start address of st3 array

.string "AVR Assembly"  ;same as asciz

str3End:           ;(past) end of str3 array

.org  0x100         ;places code at a specific location (this is sufficient)

.global main        ;eliminate (artificial) Arduino C requirement of setup & loop functions

main:

 ser   util         ;prepare to set ALL port bits for output

 out   DDR,util     ;set ddr pins for output

 ldi   XL,lo8(str3Start)   ;set 16-bit X register to point to the start of array

 ldi   XH,hi8(str3Start)   ;this requires 2 statements for the low and high bytes

 ldi   YL,lo8(str3End)     ;set 16-bit Y register to point to the start of array

 ldi   YH,hi8(str3End)     ;this requires 2 statements for the low and high bytes

again:

 movw  Z,X                 ;use 2-byte move to position Z as array index pointer

next:

;use post-increment mode to acquire byte from current index and advance the pointer

 lpm   util,Z+

 out   PORT,util           ;place the byte on the output port

 rcall delay1s            ;admire

 cp    ZL,YL               ;are we at the end of the array (Z==Y?)

 brne  next                ;if not, continue to next byte in the array

 rjmp  again               ;if so, start over, do continuously

 ret                       ;return

delay1s:

; Assembly code auto-generated

; by utility from Bret Mulvey

; Delay 8 000 000 cycles

; 1s at 8.0 MHz

   ldi  r18, 41

   ldi  r19, 150

   ldi  r20, 128

L1: dec  r20

   brne L1

   dec  r19

   brne L1

   dec  r18

   brne L1

   ret