

// Purpose :To demonstrate communication of data to the Serial Monitor in Assembly
// Results :Currently runs a SRAM dump of the GP, IO and Extended Registers
// Reference1 :https://hekilledmywire.wordpress.com/2011/01/05/using-the-usartserial-tutorial-part-2/
// Reference2 :Page 250 of the ATmega328p datasheet (for data size modes)
// Reference3 :http://nerdralph.blogspot.ca/2013/12/writing-avr-assembler-code-with-arduino.html
// Reference4 :http://www.h-renrew.de/h/avrdump/avrdump.html
// Reference5 :http://maxembedded.com/2013/09/the-usart-of-the-avr/
// Authors :C. D'Arcy (inspired by M. Blum's Medium ISP Research)
// Date :2017 02 19
// Status :Working
#include <avr/io.h>
#include <ASCII.h>
#define BAUD9600 103 //(((F\_CPU / (BAUDRATE \* 16))) - 1)
#define data 65 //ASCII value translates to 'A' or '\A
// define some null-terminated strings
label:
.asciz "This is a label..." //
gpLabel:
.asciz "32 GP Registers:\n"
ioLabel:
.asciz "64 IO Registers:\n"
exLabel:
.asciz "160 Extended Registers: "

.global setup
setup:
 ldi r25,BAUD9600 >> 8 ; pass the preferred BAUD rate to the initUART function
 ldi r24,BAUD9600 ;
 call initUART ;

 ldi ZH,hi8(gpLabel) ; obtain the address of the General Purpose Register label
 ldi ZL,lo8(gpLabel) ;
 call printStr ; transmit it...
 call printGPRegs ; transmit contents of GP Registers in a readable format
 call printLn ;

 ldi ZH,hi8(ioLabel) ; obtain the address of the Input/Output Register label
 ldi ZL,lo8(ioLabel) ;
 call printStr ; transmit it
 call printIORegs ; transmit contents of IO Registers in a readable format
 call printLn ;
 ldi ZH,hi8(exLabel) ; obtain the address of the Extended IO Register label
 ldi ZL,lo8(exLabel) ;
 call printStrLn ; transmit it
 call printExRegs ; transmit contents of Ext. IO Registers in readable format
 call printLn ;
 //call breakPoint ;

 ldi r25,data //prep ASCII character for transmission
 call printChar // TX
 call printCharLn // TX
 call printLn
 ldi r25,data+1
 call printCharLn // TX
 ldi ZH,hi8(label)
 ldi ZL,lo8(label)
 call printStr
 ldi r25,colon
 call printChar
 ldi r25,space
 call printChar
 ldi ZH,hi8(label)
 ldi ZL,lo8(label)
 call printStrLn
 ldi r23,255
again:
 mov r25,r23
 call printByte
 ldi r25,'\t
 call printChar
 mov r25,r23
 call printHex
 call printLn
 dec r23
 breq done
 rjmp again
 done:
ret

.global loop
loop:
rjmp loop //Avoid returning to the C driver