Lab 1

Reyna, Ricardo

Section 75C9

05/28/2015

* Prelab Questions:

1. What is the difference between program and data memory?

*Data memory is where we have the variables. Values are read and written into the data memory. Program memory, is where we store the program, we store constants.*

2. What memory location should the .DSEG section should start? Why?

*0x002000, because the data segment stores values in the SRAM and that's where it's located.*

3. What registers can we use to read from the program memory (flash)?

*Register Z.*

* Problems Encountered:

Instead of using lpm I used ld operation so the contents f the table wasn't loading since it's from the program memory that I have to load

* Future Work/Applications:

This lab gave as an introduction to assembly which will be useful for later labs. Assembly is used today for special applications and it will be useful for our microprocessor.

* Pseudo code:

Load Table1 with .db

Table 1 starts at 0x6370->X

Table 2 starts at 0x2C70->Y

LOOP: Load content of table1 into a register

Increase X

If content equals 0x00 branch to NULL

check if less than 0x30

if true branch to LOOP

check if greater or equal than 0x7A

if true branch to LOOP

store content into Y

increase Y

rjump to LOOP

NULL: store content into Y

increase Y

DONE: Rjump to DONE

* Program Code:

/\*

\* Lab1\_b\_RSR.asm

\*

\* Created: 5/21/2015 11:36:21 AM

\* Author: Ricardo Stefano Reyna

Lab 1 Part b

Name: Ricardo Stefano Reyna

Section#: 75C9

TA: Khaled Hassan

Description: This program grabs elements from table1 where the hex code is between 0x30 to 0x7A

\*/

.NOLIST

.include "ATxmega128A1Udef.inc"

.LIST

.ORG 0x0000

rjmp MAIN ;Go to the logic code

.ORG 0x6370

Table1: .DB 0x23, 0x75, 0x25, 0x26, 0x50, 0x5F, 0x77, 0x7B, 0x69, 0x6C, 0x24, 0x6C, 0x5F, 0x62, 0x25, 0x65, 0x5F, 0x2A, 0x33, 0x7E, 0x37, 0x2A, 0x34, 0x2F, 0x34, 0x00 ;Table with random ASCII char

.DSEG

.ORG 0x2C70

Table2: .byte 19 ;Location where the solution table is

.CSEG

.ORG 0x200

MAIN:

ldi ZL, low(Table1 << 1) ;load low bits of the table in Z

ldi ZH, high(Table1 << 1) ;load high bits of the table in Z

ldi YL, low(Table2) ;load low bits of the table in Y

ldi YH, high(Table2) ;load high bits of the table in Y

LOOP:

lpm r16, Z+ ;Load content of table and increment pointer

cpi r16, 0 ;Check if it equals 0

breq NULL ;If true go to NULL

cpi r16, 0x30 ;Check if it's less than 0x30

brlt LOOP ;If true go to LOOP

cpi r16, 0x7A ;Check if it's greater or qual than 0x7A

brge LOOP ;If true go to LOOP

st Y+, r16 ;Store in Y and increment pointer

rjmp LOOP ;Go to LOOP

NULL:

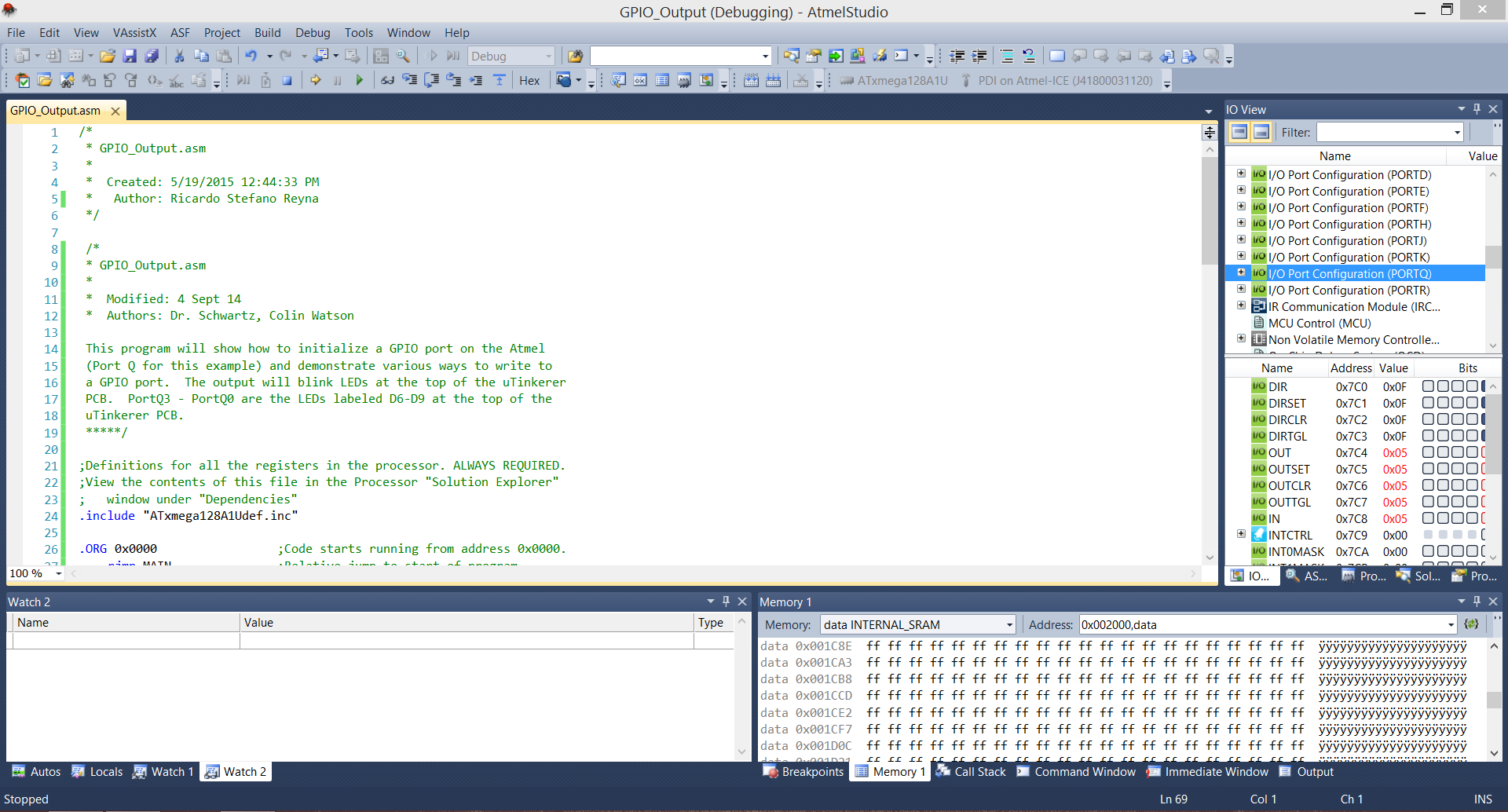
st Y+, r16 ;Store into Y and increment pointer

DONE:

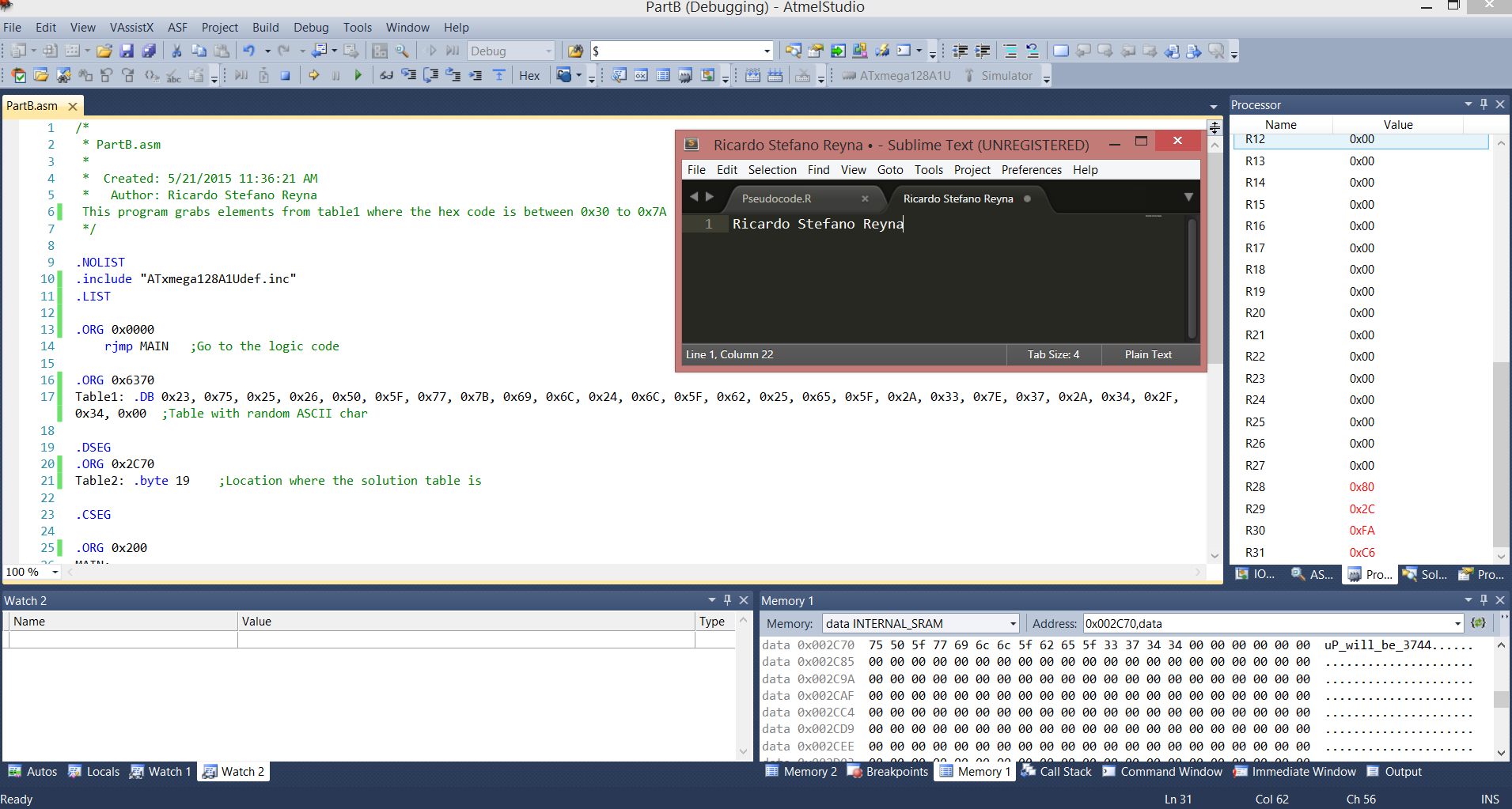
rjmp Done ;Infinite loop

* Appendix:

PartA



PartB:



PartC:  
