;-------------------------------------------------------------------------------

; Name: Evan Richter

; Term: Fall 2014

; MCU: MSP430G2553

; Assignment: 3

; Date: 10 September 2014

; Documentation: MSP430 CPU Instruction Set

; "Peasant Multiplication" from English Wikipedia

; no cadet help

; Functionality: A, B, and basic functionalities achieved

;-------------------------------------------------------------------------------

.cdecls C,LIST,"msp430.h" ; BOILERPLATE

**.text** ; BOILERPLATE

.retain ; BOILERPLATE

.retainrefs ; BOILERPLATE

**.global** main ; BOILERPLATE

**plus:** .equ 0x11

**minus:** .equ 0x22

**times:** .equ 0x33

**clear:** .equ 0x44

**end:** .equ 0x55

**input:** **.byte** 0x22, 0x11, 0x22, 0x22, 0x33, 0x33, 0x08, 0x44, 0x08, 0x22, 0x09, 0x44, 0xff, 0x11, 0xff, 0x44, 0xcc, 0x33, 0x02, 0x33, 0x00, 0x44, 0x33, 0x33, 0x08, 0x55

;-------------------------------------------------------------------------------

; main: steps through input string with a case switch

;

; R5: array pointer to unused byte

; R6: accumulator

; R7: memory pointer to available byte

;-------------------------------------------------------------------------------

**main:**

**mov.w** #\_\_STACK\_END,SP ; BOILERPLATE Initialize stackpointer

**mov.w** #WDTPW|WDTHOLD,&WDTCTL ; BOILERPLATE Stop watchdog timer

**mov.w** #0x00, R6

**mov.w** #0x0200, R7

**mov.w** #input, R5

**jmp** enternum ; first value always is number not operation

**checkend:**

**cmp.b** #end, 0(R5)

**jnz** checkplus

**jmp** trapcpu

**checkplus:**

**cmp.b** #plus, 0(R5)

**jnz** checkminus

**inc.w** R5

**add.b** @R5+, R6

**jnc** nocarry

**mov.w** #0xFF, R6 ; B functionality

**nocarry:**

**jmp** save

**checkminus:**

**cmp.b** #minus, 0(R5)

**jnz** checktimes

**inc.w** R5

**cmp.b** @R5, R6 ; R6 >= R5?

**jge** notnegative ; yes: goto notnegative

**mov.w** #0x00, R6

**inc.w** R5

**jmp** save

**notnegative:**

**sub.b** @R5+, R6

**jmp** save

**checktimes:** ; multiplication using Peasant Multiplication method

**cmp.b** #times, 0(R5)

**jnz** checkclear

**inc.w** R5

**cmp.b** #0x00, 0(R5)

**jz** zero

**push.w** R7 ; preserve R7

**mov.w** #0x00, R7

**mov.b** @R5+, R7

**mov.w** #0x00, R8 ; R8 holds the final answer as R6 doubles

**checkdone:**

**cmp.w** #0x00, R7

**jz** done

**bit.b** #0x1, R7 ; checks even-ness of R7

**jz** even

**add.w** R6, R8

**even:**

**rra.w** R7 ; R7 halves

**rla.w** R6 ; R6 doubles

**jmp** checkdone

**done:**

**pop.w** R7

**cmp.w** #0x100, R8 ; R8 < 0x100?

**jl** goodmult ; yes: goto goodmult

**mov.w** #0xFF, R8 ; no: product was too big

**goodmult:**

**mov.w** R8, R6

**jmp** save

**zero:**

**inc.w** R5

**mov.b** #0x00, R6

**jmp** save

**checkclear:**

**cmp.b** #clear, 0(R5)

**jnz** enternum

**mov.b** #0x00, 0(R7)

**inc.w** R7

**inc.w** R5

**mov.b** @R5+, R6

**jmp** checkend

**enternum:**

**mov.b** @R5+, R6

**jmp** checkend

**trapcpu:**

**jmp** trapcpu

**save:**

**mov.b** R6, 0(R7)

**inc.w** R7

**jmp** checkend

;-------------------------------------------------------------------------------

; System Initialization

;-------------------------------------------------------------------------------

**.global** \_\_STACK\_END ; BOILERPLATE

**.sect** .stack ; BOILERPLATE

**.sect** ".reset" ; BOILERPLATE MSP430 RESET Vector

**.short** main ; BOILERPLATE