;Initialize Stack

.ORG 0

LDI R20,HIGH(RAMEND)

OUT SPH,R20

LDI R20,LOW(RAMEND)

OUT SPL,R20

;Initialization of used registers.

CLR R15 ;Set register 15 to 0

CLR R14 ;Clearing R14 for counter 45 func()

CLR R21 ;Clear R21 for value counter needs to reach

CLR R7 ;Dummy register for Task 3

LDI R21,45 ;Load second counter value of 45

MOV R14,R21 ;Move value into R14

CLR R21 ;Clear R21 again

LDI R22,255 ;Counter

LDI R21,0 ;Counter Goal

LDI R23, 5 ;Divisor

LDI R25, 0 ;Division Counter

CLR R18 ;ZL for numbers not divisible by 5

CLR R19 ;ZH for numbers not divisible by 5

;Initiate Y Register to Address 0x0400

LDI R28, LOW(0x0400)

LDI R29, HIGH(0x0400)

;Initiate Z Register to Address 0x0600

LDI R30, LOW(0x0600)

LDI R31, HIGH(0x0600)

;Initial Addition to Reg X.

LDI R26, LOW(0x0222)

LDI R27, HIGH(0x0222)

ADD R15, R26

ADC R15, R27

ST X+, R15

DEC R22

JMP L1

;Start of DA Tasks, X Register Addition

L1: CLR R15 ;Reset R15 to 0

ADD R15, XH ;Load XH value into R15

ADC R15, XL ;Load XL value into R15

MOV R24, R15 ;Move R15 value into R24 for DIV\_BY\_5 Func()

CALL DIV\_BY\_5 ;Call DIVIDE\_BY\_5 subroutine

ST X+, R15 ;Store R15 value into memory pointed by X Register

CP R22, R21 ;Compare Counter

BRNE LOOP ;Branch to LOOP if counter is not equal to 0

CP R22,R21

BREQ AFTER\_ZERO

;Counter Decrement Function()

LOOP: DEC R22 ;Decrement counter by one

JMP L1 ;Jump back to L1 label

;Function to count down 45 more times for a total of 300

AFTER\_ZERO: DEC R14 ;Decrement counter containing 45

CP R14,R21 ;Compare counter w/ 45 to 0

BRNE L1 ;Branch back to L1 label if counter is not equal to 0

JMP RESET\_COUNTER ;Jump to label to reset all counter registers

;Function for checking if R15 value is divisible by 5

DIV\_BY\_5: INC R25 ;Increment division counter by one

SUB R24, R23 ;Subtract Dividend (R24) by Divisor (R23)

BRCC DIV\_BY\_5 ;BRCC back to start of function until carry (end of subtraction) is detected

DEC R25 ;Decrement division counter by one

ADD R24, R23 ;Add values of Dividend and Divisor for quotient

CP R24, R21 ;Compare Dividen with 0

BRNE Z\_NOT\_EQ\_0 ;Branch to SUB\_5 Func() if Dividend does not equal 0

CP R24,R21 ;Compare dividend to 0 again for BREQ instruction

BREQ Z\_EQ\_0 ;Branch to Z\_EQ\_0 func()

RET ;Return to L1

;Function used to store the value of R15 into the Z register, incrememnt memory address, then return to L1

Z\_EQ\_0: ST Z+,R15 ;Store value of R15 into Z register then change Z pointer address

CLR R24 ;Clear Dividend register value

RET ;Return to L1

;Function used to load results of R24 if not equal to 0 (Divisible by 5)

Z\_NOT\_EQ\_0: ;ADD R16, HIGH(R15) ;Add the high value of R15 to R16

;ADC R17, LOW(R15) ;Add the low value of R15 to R17

ST Y+, R15 ;Store R15 value into Y register then change Y pointer address

CLR R24 ;Reset R24 to 0

RET ;Return to L1

;Function used to reset the counter registers after main X register addition and register Y & Z have been filled up

RESET\_COUNTER: CLR R22 ;Clear register for counter value 255

LDI R22,255 ;Load value of 255 into register

CLR R21 ;Clear register to load 45

LDI R21,45 ;Load the value of 45 into register

CLR R14 ;Clear R14

MOV R14,R21 ;Move value of 45 into R14 from R21

CLR R21 ;Clear R21 for use as counter objective

JMP Z\_AND\_Y ;Jump to Z\_AND\_Y to start adding Y and Z reg values into R16:R17, and R18:R19 respectively

;Function used to initiate X/Y/Z registers for Task 3. Set memory addresses back to their start then add the initial values into their respective registers

;Takes every to memory locations of Y and Z, adds them into X, then places High and Low values into respective registers (r16,17,18,19)

Z\_AND\_Y: LDI R28, HIGH(0x0400) ;Point Y to 0x400

LDI R29, LOW(0x0400) ;Point Y to 0x400

LDI R30, HIGH(0x0600) ;Point Z to 0x0600

LDI R31, LOW(0x0600) ;Point Z to 0x0600

LD R7,X+ ;Load a value to simply increment address of X reg by 1

CLR R26 ;Clear contents of X

CLR R27 ;Clear contents of X

;Add Z reg values into X

ADD R26,ZH

ADC R27,ZL

;Increment memory of Z

LD R7,Z+

;Add Z reg values into reg X again

ADC R26,ZH

ADC R27,ZL

;Place Z sums into R18 and R19

ADD R18, XH

ADC R19, XL

;Reset X reg value to 0

CLR R26

CLR R27

;Add contents of Y into X

ADD R26,YH

ADC R27,YL

;Increment Y memory address

LD R7,Y+

;Add contents into X again

ADC R26,YH

ADC R27,YL

;Place resulting sums into R16,R17

ADD R16,XH

ADC R17,XL

;Decrememnt counter by 1

DEC R22

;Jump to label to continue process 299 more times

JMP CYCLE\_ZY

;This function is to continue the previous process of placing Y & Z sums into their respective registers 299 more times

CYCLE\_ZY: CLR R26 ;Reset X

CLR R27 ;Reset X

ADD R26,ZH

ADC R27,ZL

LD R7,Z+

ADD R26,ZH

ADC R27,ZL

ADD R18, XH

ADC R19, XL

CLR R26

CLR R27

ADD R26,YH

ADC R27,YL

LD R7,Y+

ADD R26,YH

ADC R27,YL

ADD R16,XH

ADC R17,XL

;Portion of code is to cycle through the same process 255 times first then 45 more times for a total of 300

CP R22,R21

BRNE COUNTER\_255

CP R14,R21

BRNE COUNTER\_45

JMP END

;Function to keep track of counter with 255, jumps back to CYCLE\_ZY

COUNTER\_255: DEC R22

JMP CYCLE\_ZY

;Function to keep track of counter with 45, jumps back to CYCLE\_ZY

COUNTER\_45: DEC R14

JMP CYCLE\_ZY

;End of program

END: NOP

JMP END