CPE301 – FALL 2019

Design Assignment 1B

Student Name: Dillon Archibald
Student #: 5004439916
Student Email: archid1@unlv.nevada.edu
Primary Github address: https://github.com/Dil-bert/Alabaster.git
Directory: Alabaster

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

Atmel studio 7

List of Components used

Block diagram with pins used in the Atmega328P

2. INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/A

N/A

3. DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/A

```
;
; AssemblerApplication1.asm
;
; Created: 9/21/2019 7:09:45 PM
; Author : Dilbert
;
.include <m328pbdef.inc>
;DEFINING GLOBAL CONSTANTS;
.EQU DIVIS                      = 0x05          ;WHAT IS BEING DIVIDED BY
.EQU QUANTITY           = 0xFA          ;HOW MANY ARE BEING CREATED
.EQU MAXQUANT           = 0xFF          ;MAX VALUE POSSIBLE

.SET STARTADDS          = 0x0200        ;MEMORY ADDRESS FOR INITIAL FILL
.SET DIVISABLE          = 0x0300        ;MEMORY ADDRESS FOR VALUES DIVISIBLE BY FIVE
.SET NONDIVISABLE       = 0x0500        ;MEMORY ADDRESS FOR VALUES NOT DIVISIBLE BY FIVE

;THIS MACRO WILL CHECK FOR DIVISIBILITY BY FIVE, AND ONLY FIVE.
.MACRO IS_DIVIS                                         ;MACRO FOR DETERIMINING IF A VALUE IS
DIVISABLE BY FIVE
DIVISABATOR:                                            ;START OF LOOP
SUBI @0,5                                               ;SUBTRACT 5 FROM THE DIVIDEND
BREQ CHICKEN_DINNER                             ;IF THAT BRINGS THE DIVIDEND TO ZERO, YOU HAVE A
WINNER, GO TO CHICKEN DINNER
MOV R16,@0                                              ;COPY THE VALUE IN THE PROVIDED REGISTER TO
R16
SUBI R16,4                                              ;SUBTRACT 4 FROM THE DIVIDEND IN R16
BREQ JIM_CARRY                                  ;IF EQUAL TO ZERO, IT'S A LOOSAHER (LOSER) BREAK TO
JIM_CARRY
MOV R16,@0                                              ;COPY THE VALUE IN THE PROVIDED REGISTER TO
R16 AGAIN (RESET)
SUBI R16,3                                              ;SUBTRACT 3 FROM THE DIVIDEND IN R16
```

```
BREQ JIM_CARRY                          ;IF EQUAL TO ZERO, IT'S A LOOSAHER (LOSER) BREAK TO
JIM_CARRY
MOV R16,@0                                      ;COPY THE VALUE IN THE PROVIDED REGISTER TO
R16 AGAIN (RESET)
SUBI R16,2                                      ;SUBTRACT 2 FROM THE DIVIDEND IN R16
BREQ JIM_CARRY                          ;IF EQUAL TO ZERO, IT'S A LOOSAHER (LOSER) BREAK TO
JIM_CARRY
MOV R16,@0                                      ;COPY THE VALUE IN THE PROVIDED REGISTER TO
R16 AGAIN (RESET)
SUBI R16,1                                      ;SUBTRACT 1 FROM THE DIVIDEND IN R16
BREQ JIM_CARRY                          ;IF EQUAL TO ZERO, IT'S A LOOSAHER (LOSER) BREAK TO
JIM_CARRY
RJMP DIVISABATOR                        ;IF FAILD ALL THESE CHECKS, JUMP BACK UP TO TOP OF
LOOP
JIM_CARRY:                                      ;WHERE NON-DIVISABLE BY FIVE NUMBERS
GO
LDI R16, 0x00                           ;LOAD ZERO INTO R16 (ACTS AS A BOOLEAN FALSE)
RJMP ENDERMAKER                         ;JUMP TO THE END OF MACRO
CHICKEN_DINNER:                                 ;WHERE NUMBERS THAT ARE DIVISABLE BY FIVE GO
LDI R16, 0x01                           ;LOAD ONE INTO R16 (ACTS AS A BOOLEAN TRUE)
ENDERMAKER:                                     ;END LABEL TO BE JUMPED TO BY JIM-
CARRY
.ENDMACRO

;************ MAIN PROG BEGIN ********************
LDI XL, LOW(STARTADDS)                  ;LOADING THE STARTADDS MEMORY ADDRESS INTO THE LOW SIDE OF X
REGISTER
LDI XH, HIGH(STARTADDS)                         ;LOADING THE STARTADDS MEMORY ADDRESS INTO THE HIGH
SIDE OF X REGISTER
LDI YL, LOW(DIVISABLE)                  ;LOADING THE DIVISABLE MEMORY ADDRESS INTO THE LOW SIDE OF Y
REGISTER
LDI YH, HIGH(DIVISABLE)                         ;LOADING THE DIVISABLE MEMORY ADDRESS INTO THE HIGH
SIDE OF Y REGISTER
LDI ZL, LOW(NONDIVISABLE)               ;LOADING THE NONDIVISABLE MEMORY ADDRESS INTO THE LOW SIDE
OF THE Z REGISTER
LDI ZH, HIGH(NONDIVISABLE)              ;LOADING THE NONDIVISABLE MEMORY ADDRESS INTO THE HIGH SIDE
OF THE Z REGISTER
LDI R25, DIVIS                                  ;LOADING THE DIVIS VALUE (0x05) INTO R25
CALL FILL_VALS                          ;CALLING THE FUNCTION FILL_VALS
POST:                                           ;LABEL POST: RETURN POINT OF
BADDLY_DIVIS AND GOODLY_DIVIS
LD R24,X+                                       ;LOAD THE VALUE STORED AT LOCATION POINTED TO
BY X, INTO REGISTER R24, THE INCREMENT POINTER VALUE
CPI R24, 0x00                           ;COMPARE THIS VALUE TO ZERO IN ORDER TO DETERMIN IF
THE END OF THE SET HAS BEEN REACHED
BREQ NEXT                                       ;IF (R24 == 0) BRANCH TO THE LOCATION "NEXT;"
MEANING THE ORGINIZATION PHAZE IS OVER
MOV R23,R24                                     ;COPY R24 INTO R23 IN ORDER TO PRESERVE THE
VALUE HELD BY R24 WHEN CALLING IS_DIVIS MACRO
IS_DIVIS R23                            ;CALLING IS_DIVIS MACRO
CPI R16, 0x00                           ;COMPARE R16 (BOOLEAN 0/1) WITH ZERO
BREQ BADDLY_DIVIS                       ;BRANCH IF R16 RETURNS AS ZERO (VALUE IN R24 IS NOT
DIVISABLE BY 5)
BRNE GOODLY_DIVIS                       ;BRANCH IF R16 RETURNS AS ONE (VALUE IN R24 IS
DIVISBALE BY 5)
JMP ERROR                                       ;THIS INSTRUCTION SHOULD NEVER BE REACHED, IF
REACHED THERE IS AN ERROR
;       JMP ERROR IS ACTING AS A "DEFAULT CASE STATMENT"
```

```
;COMPARISONS ARE COMPLETE, NOW TO DO PORTION 3 OF DA1B
;(UNSURE OF WHAT THE INSTRUCTIONS WERE ASKING FOR SO THIS IS MY INTERPRITATION OF THE
INSTRUCTIONS)
NEXT:
SUB R17,R17                                          ;ENSURE R17 IS ZERO
SUB R19,R19                                          ;ENSURE R19 IS ZERO
SUB R20,R20                                          ;ENSURE R20 IS ZERO
LDI YL, LOW(DIVISABLE)                 ;RELOAD REG YLOW WITH LOW OF DIVISABLE (STARTING LOCATION OF
THE DIVISABLE DATA)
LDI YH, HIGH(DIVISABLE)                        ;RELOAD REG YHIGH WITH HIGH OF DIVISABLE (STARTING
LOCATION OF THE DIVISABLE DATA)
LDI ZL, LOW(NONDIVISABLE)              ;RELOAD REG ZLOW WITH LOW OF NONDIVISABLE (STARTING LOCATION
OF THE NONDIVISABLE DATA)
LDI ZH, HIGH(NONDIVISABLE)             ;RELOAD REG ZHIGH WITH HIGH OF NONDIVISABLE (STARTING
LOCATION OF THE NONDIVISABLE DATA)
LD R16,Z+                                            ;LOAD THE VALUE POINTED TO BY Z INTO R16, AND
THEN INCREMENT THE POINTER VALUE
LD R22,Y+                                            ;LOAD THE VALUE POINTED TO BY Y INTO R22, AND
THEN INCREMENT THE POINTER VALUE
ADD R16,R22                                          ;ADD R16 AND R22 VALUES AND STORE THEM INTO
R16
ADC R17,R20                                          ;ADD R17(ZERO) WITH R20(ALSO ZERO) AND THE
CARRY, STORE INTO R17
LD R18,Z                                             ;LOAD THE VALUE POINTED TO BY Z INTO R18
LD R22,Y                                             ;LOAD THE VALUE POINTED TO BY Y INTO R22
ADD R18,R22                                          ;ADD R18 AND R22 VALUES AND STORE THEM INTO
R18
ADC R19,R20                                          ;ADD R19(ZERO) WITH R20(ALSO ZERO) AND THE
CARRY, STORE INTO R19
DONE:
JMP DONE                                             ;LOOP DONE

ERROR:                                                       ;ERROR STATE SHOULD NEVER BE REACHED,
IF IT IS IT WILL STAY IN INFINATE LOOP FORCING REBOOT?
JMP ERROR

BADDLY_DIVIS:                                        ;IF THE RESULT OF IS_DIVIS IS 0, THEN R24 IS
NOT DIVISABLE BY FIVE, PROGRAM JUMPS HERE
ST Z+,R24                                            ;STORE THE VALUE IN R24 IN THE MEMORY
LOCATION POINTED TO BY Z, THEN INCREMENT Z
RJMP POST                                            ;JUMP BACK TO THE MAIN PROGRAM AT POST:

GOODLY_DIVIS:                                        ;IF THE RESULT OF IS_DIVIS IS 1, THEN R24 IS
DIVISABLE BY FIVE, PROGRAM JUMPS HERE
ST Y+,R24                                            ;STORE THE VALUE IN R24 IN THE MEMORY
LOCATION POINTED TO BY Y, THEN INCREMENT Y
RJMP POST                                            ;JUMP BACK TO THE MAIN PROGRAM AT POST

FILL_VALS:                                                   ;FILL THE MEMORY WITH DATA
LDI R16, MAXQUANT                          ;THIS STARTS AS THE MAXIMUM NUMBER THAT A VALUE CAN
BE, AND IS THEN DECREMENTED UNTIL THE QUANTITY IS ZERO
LDI R18, QUANTITY                          ;THIS IS HOW MANY NUMBERS ARE LEFT TO BE GENERATED
LOOP:
ST X+,R16                                            ;STORE THE VALUE IN R16 TO THE MEMORY
LOCATION POINTED TO BY X, THEN INCREMENT THE POINTER VALUE
DEC R16                                              ;DECREMENT THE VALUE STORED IN R16
DEC R18                                              ;DECREMENT THE VALUE STORED IN R18
BRNE LOOP                                            ;IF R18 IS NOT ZERO, BRANCH BACK UP TO LOOP
```

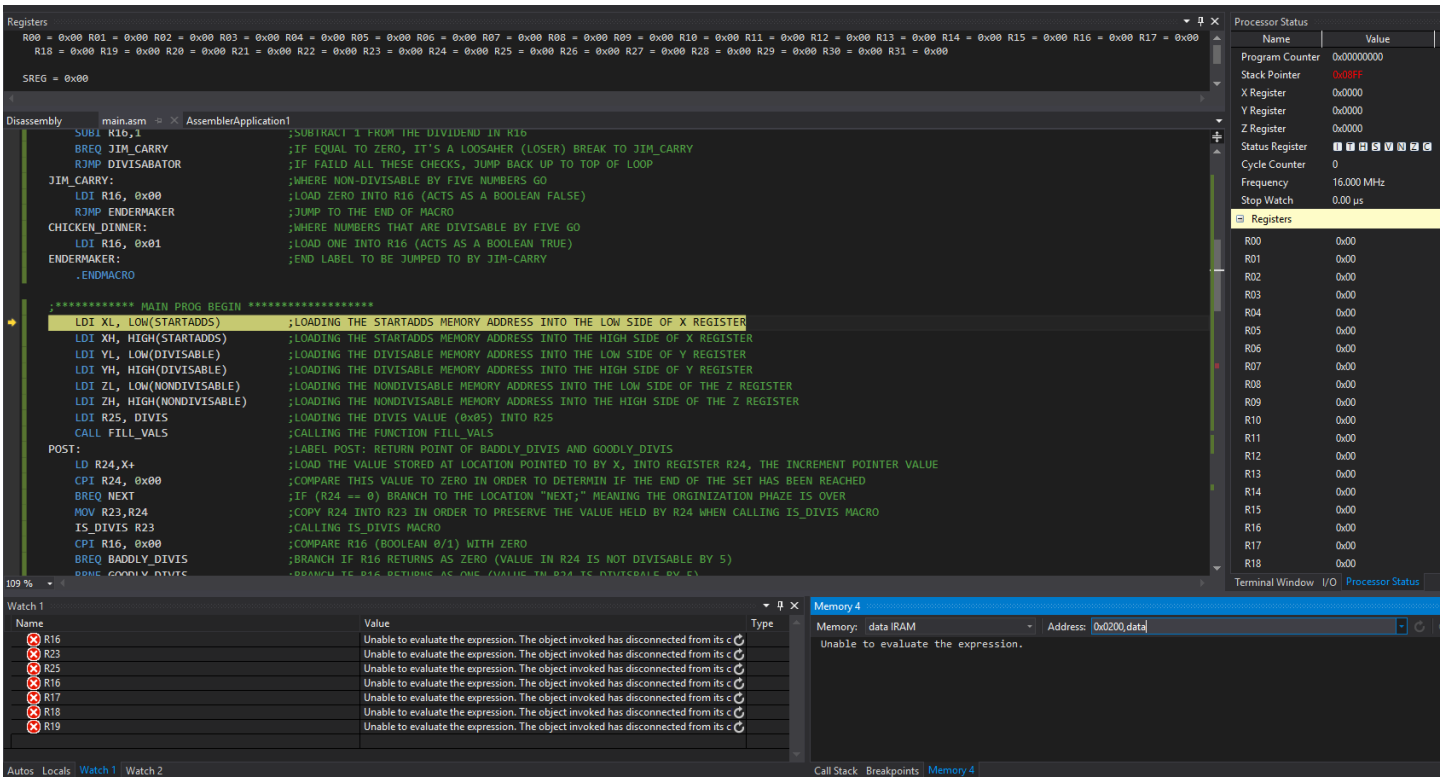
```
LDI XL, LOW(STARTADDS)                  ;RESTORE THE POINTER VALUE OF REG X TO BE REUSED LATER
LDI XH, HIGH(STARTADDS)                         ;RESTORE THE POINTER VALUE OF REG X TO BE REUSED
LATER IN PROGRAM
RET


;******** UNUSED BUT KEPT FOR FUTURE CONSIDERATION **********
;-----MADE IN AN ATTEMPT TO CREATE REUSABLE CODE FOR FUTURE APLICATIONS--------
;THIS MACRO WILL EMPTY ARG 0, BE SURE YOU CAN LOSE WHAT IS SENT TO ME
;WILL CHECK IF ONE NUMBER IS DIVISABLE BY ANY OTHER NUMBER
;RETURNS
;SYNTAX- "IS_DIVIS REG_DIVIDEND, REG_DIVISOR"
;.MACRO IS_DIVIS
;DIVISABATOR:
;       SUB @0, @1
;       BREQ CHICKEN_DINNER
;       BRMI JIM_CARRY
;       RJMP DIVISABATOR
;JIM_CARRY:
;       LDI R16, 0x00
;       RJMP ENDERMAKER
;CHICKEN_DINNER:
;       LDI R16, 0x01
;ENDERMAKER:
;       .ENDMACRO
;-------------------------------------------------------------------------------
```

4. SCHEMATICS

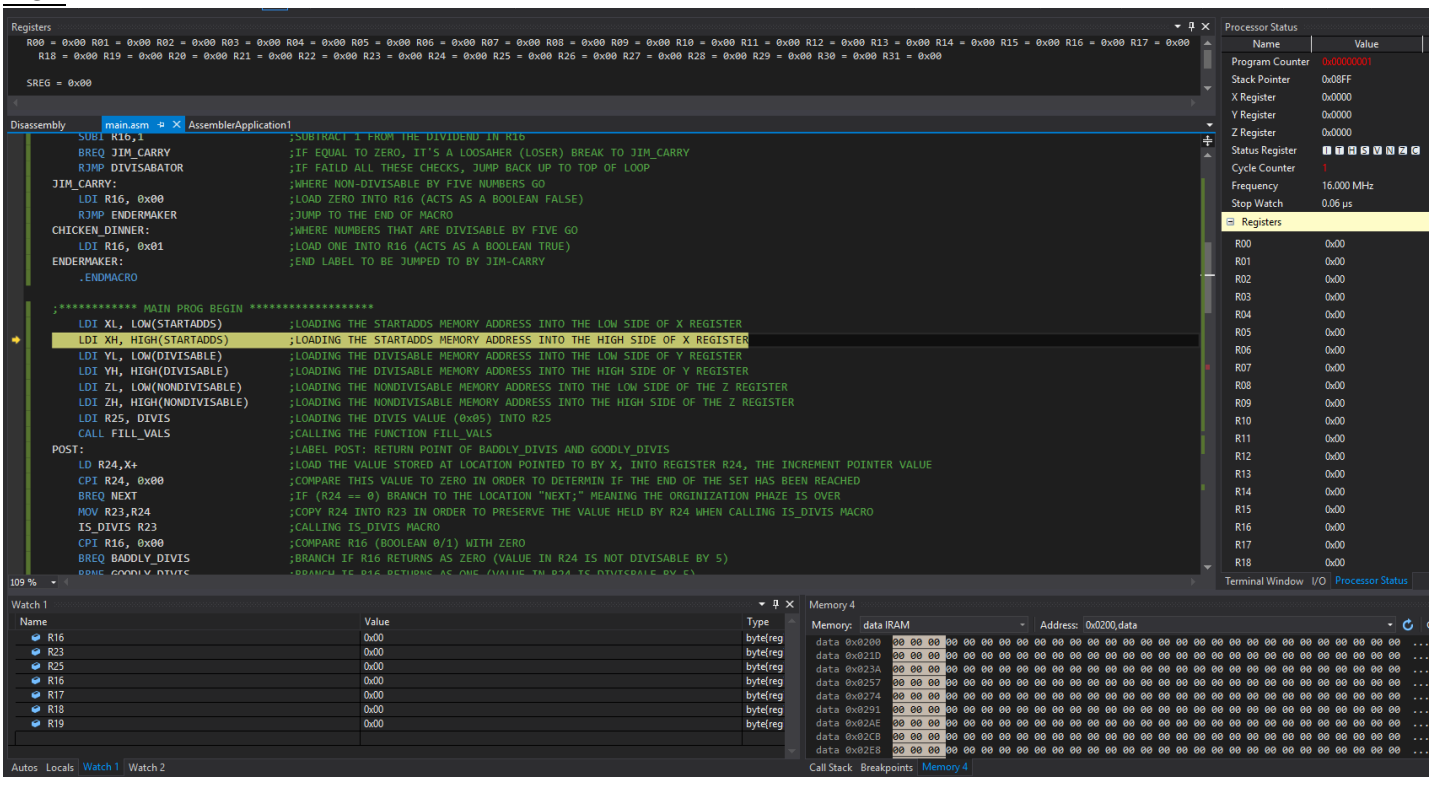
Use fritzing.org

5. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

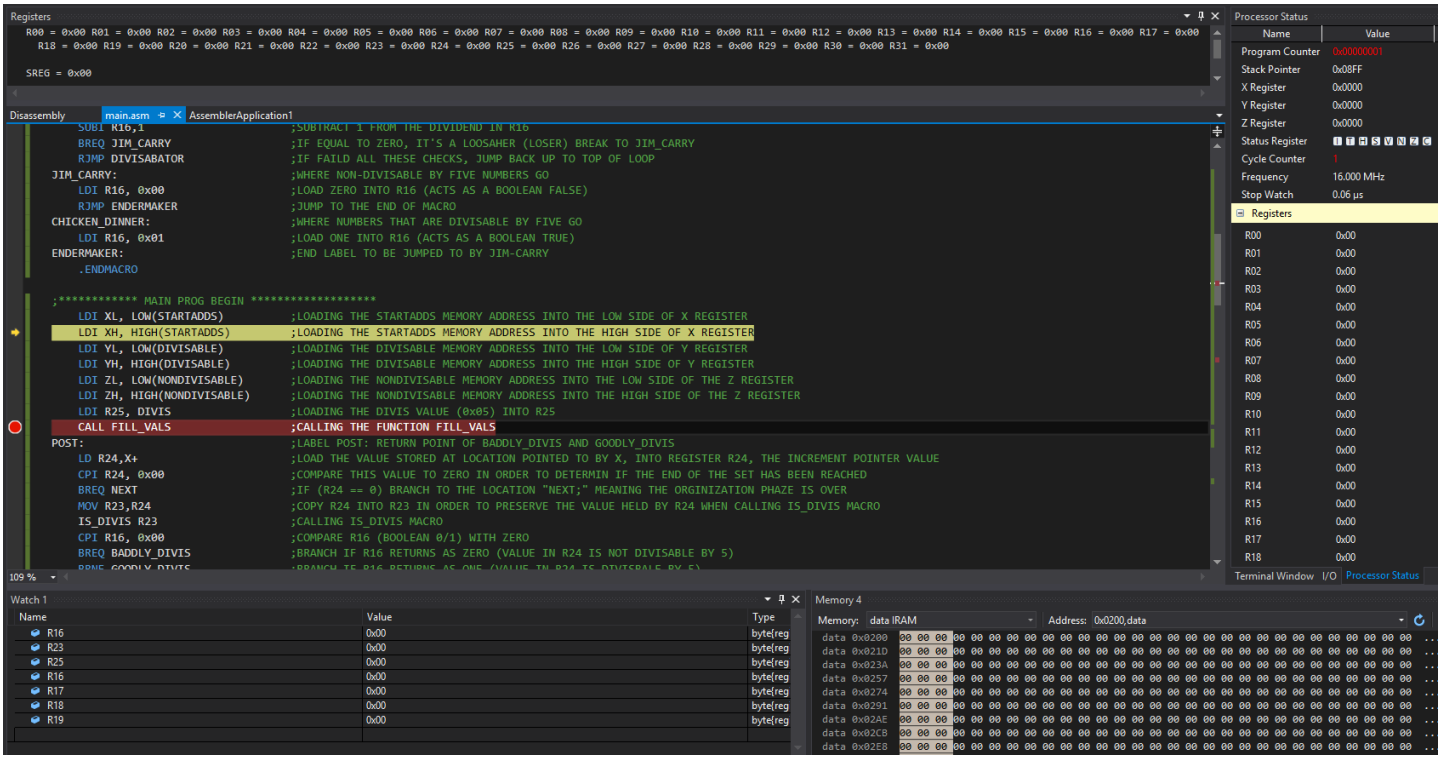
Pre 1


Registers
R00 = 0x00 R01 = 0x00 R02 = 0x00 R03 = 0x00 R04 = 0x00 R05 = 0x00 R06 = 0x00 R07 = 0x00 R08 = 0x00 R09 = 0x00 R10 = 0x00 R11 = 0x00 R12 = 0x00 R13 = 0x00 R14 = 0x00 R15 = 0x00 R16 = 0x00 R17 = 0x00
R18 = 0x00 R19 = 0x00 R20 = 0x00 R21 = 0x00 R22 = 0x00 R23 = 0x00 R24 = 0x00 R25 = 0x00 R26 = 0x00 R27 = 0x00 R28 = 0x00 R29 = 0x00 R30 = 0x00 R31 = 0x00
SREG = 0x00
Disassembly
main.asm
AssemblerApplication1
SUBI R16,1 ;SUBTRACT 1 FROM THE DIVIDEND IN R16
BREQ JIM_CARRY ;IF EQUAL TO ZERO, IT'S A LOOSAHER (LOSER) BREAK TO JIM_CARRY
RJMP DIVISABATOR ;IF FAILD ALL THESE CHECKS, JUMP BACK UP TO TOP OF LOOP
JIM_CARRY: ;WHERE NON-DIVISABLE BY FIVE NUMBERS GO
LDI R16, 0x00 ;LOAD ZERO INTO R16 (ACTS AS A BOOLEAN FALSE)
RJMP ENDERMAKER ;JUMP TO THE END OF MACRO
CHICKEN_DINNER: ;WHERE NUMBERS THAT ARE DIVISABLE BY FIVE GO
LDI R16, 0x01 ;LOAD ONE INTO R16 (ACTS AS A BOOLEAN TRUE)
ENDERMAKER: ;END LABEL TO BE JUMPED TO BY JIM-CARRY
.ENDMACRO
;************ MAIN PROG BEGIN ******************
LDI XL, LOW(STARTADDS) ;LOADING THE STARTADDS MEMORY ADDRESS INTO THE LOW SIDE OF X REGISTER
LDI XH, HIGH(STARTADDS) ;LOADING THE STARTADDS MEMORY ADDRESS INTO THE HIGH SIDE OF X REGISTER
LDI YL, LOW(DIVISABLE) ;LOADING THE DIVISABLE MEMORY ADDRESS INTO THE LOW SIDE OF Y REGISTER
LDI YH, HIGH(DIVISABLE) ;LOADING THE DIVISABLE MEMORY ADDRESS INTO THE HIGH SIDE OF Y REGISTER
LDI ZL, LOW(NONDIVISABLE) ;LOADING THE NONDIVISABLE MEMORY ADDRESS INTO THE LOW SIDE OF THE Z REGISTER
LDI ZH, HIGH(NONDIVISABLE) ;LOADING THE NONDIVISABLE MEMORY ADDRESS INTO THE HIGH SIDE OF THE Z REGISTER
LDI R25, DIVIS ;LOADING THE DIVIS VALUE (0x05) INTO R25
CALL FILL_VALS ;CALLING THE FUNCTION FILL_VALS
POST: ;LABEL POST: RETURN POINT OF BADDLY_DIVIS AND GOODLY_DIVIS
LD R24,X+ ;LOAD THE VALUE STORED AT LOCATION POINTED TO BY X, INTO REGISTER R24, THE INCREMENT POINTER VALUE
CPI R24, 0x00 ;COMPARE THIS VALUE TO ZERO IN ORDER TO DETERMIN IF THE END OF THE SET HAS BEEN REACHED
BREQ NEXT ;IF (R24 == 0) BRANCH TO THE LOCATION "NEXT;" MEANING THE ORGINIZATION PHAZE IS OVER
MOV R23,R24 ;COPY R24 INTO R23 IN ORDER TO PRESERVE THE VALUE HELD BY R24 WHEN CALLING IS_DIVIS MACRO
IS_DIVIS R23 ;CALLING IS_DIVIS MACRO
CPI R16, 0x00 ;COMPARE R16 (BOOLEAN 0/1) WITH ZERO
BREQ BADDLY_DIVIS ;BRANCH IF R16 RETURNS AS ZERO (VALUE IN R24 IS NOT DIVISABLE BY 5)
109 %
Processor Status
Name
Value
Program Counter 0x00000000
Stack Pointer 0x08FF
X Register 0x0000
Y Register 0x0000
Z Register 0x0000
Status Register I T H S V N Z C
Cycle Counter 0
Frequency 16.000 MHz
Stop Watch 0.00 µs
Registers
R00 0x00
R01 0x00
R02 0x00
R03 0x00
R04 0x00
R05 0x00
R06 0x00
R07 0x00
R08 0x00
R09 0x00
R10 0x00
R11 0x00
R12 0x00
R13 0x00
R14 0x00
R15 0x00
R16 0x00
R17 0x00
R18 0x00
Terminal Window I/O Processor Status
Watch 1
Name
Value
Type
R16 Unable to evaluate the expression. The object invoked has disconnected from its c
R23 Unable to evaluate the expression. The object invoked has disconnected from its c
R25 Unable to evaluate the expression. The object invoked has disconnected from its c
R16 Unable to evaluate the expression. The object invoked has disconnected from its c
R17 Unable to evaluate the expression. The object invoked has disconnected from its c
R18 Unable to evaluate the expression. The object invoked has disconnected from its c
R19 Unable to evaluate the expression. The object invoked has disconnected from its c
Autos Locals Watch 1 Watch 2
Memory 4
Memory: data IRAM
Address: 0x0200,data
Unable to evaluate the expression.
Call Stack Breakpoints Memory 4

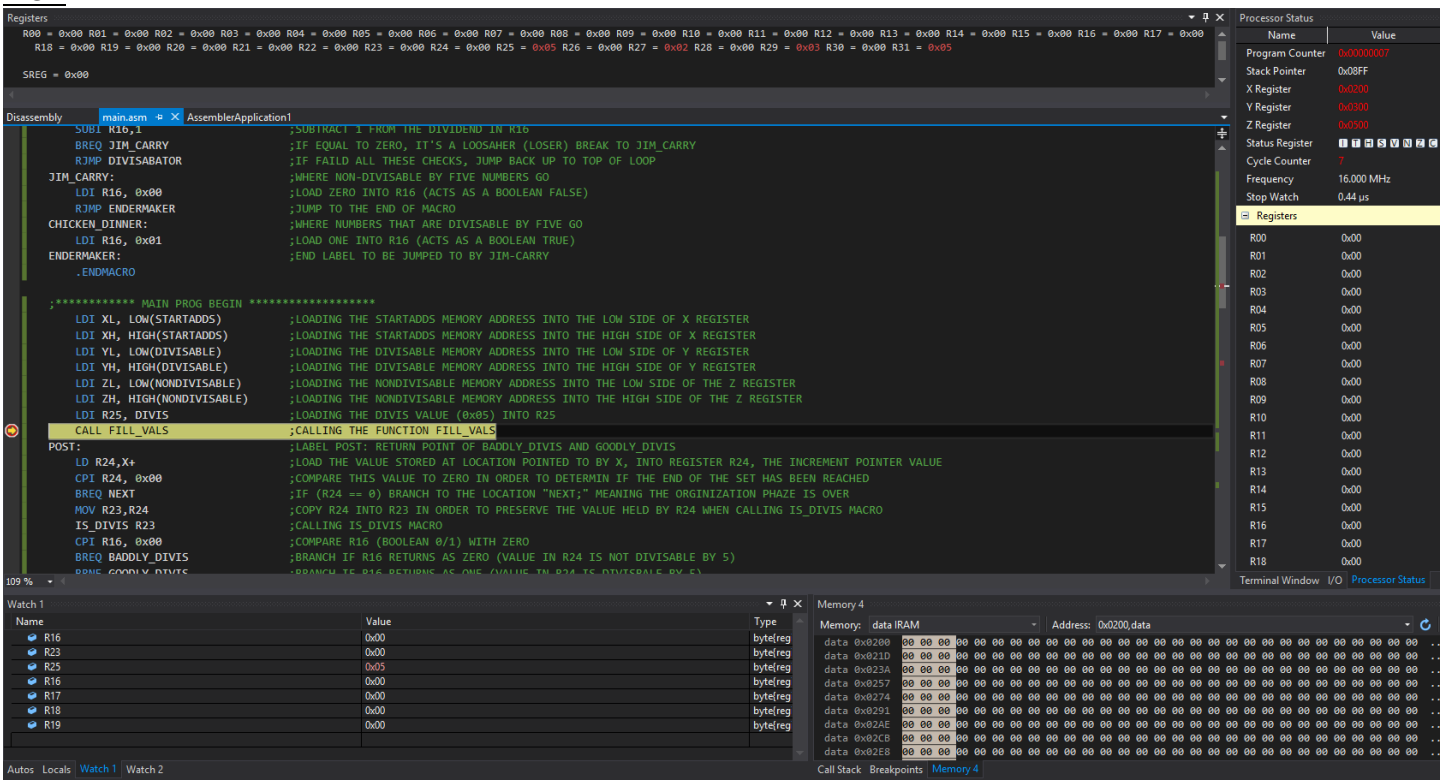
Pre 1


Registers
R00 = 0x00 R01 = 0x00 R02 = 0x00 R03 = 0x00 R04 = 0x00 R05 = 0x00 R06 = 0x00 R07 = 0x00 R08 = 0x00 R09 = 0x00 R10 = 0x00 R11 = 0x00 R12 = 0x00 R13 = 0x00 R14 = 0x00 R15 = 0x00 R16 = 0x00 R17 = 0x00
R18 = 0x00 R19 = 0x00 R20 = 0x00 R21 = 0x00 R22 = 0x00 R23 = 0x00 R24 = 0x00 R25 = 0x00 R26 = 0x00 R27 = 0x00 R28 = 0x00 R29 = 0x00 R30 = 0x00 R31 = 0x00
SREG = 0x00
Disassembly
main.asm
AssemblerApplication1
SUBI R16,1 ;SUBTRACT 1 FROM THE DIVIDEND IN R16
BREQ JIM_CARRY ;IF EQUAL TO ZERO, IT'S A LOOSAHER (LOSER) BREAK TO JIM_CARRY
RJMP DIVISABATOR ;IF FAILD ALL THESE CHECKS, JUMP BACK UP TO TOP OF LOOP
JIM_CARRY: ;WHERE NON-DIVISABLE BY FIVE NUMBERS GO
LDI R16, 0x00 ;LOAD ZERO INTO R16 (ACTS AS A BOOLEAN FALSE)
RJMP ENDERMAKER ;JUMP TO THE END OF MACRO
CHICKEN_DINNER: ;WHERE NUMBERS THAT ARE DIVISABLE BY FIVE GO
LDI R16, 0x01 ;LOAD ONE INTO R16 (ACTS AS A BOOLEAN TRUE)
ENDERMAKER: ;END LABEL TO BE JUMPED TO BY JIM-CARRY
.ENDMACRO
;************ MAIN PROG BEGIN ******************
LDI XL, LOW(STARTADDS) ;LOADING THE STARTADDS MEMORY ADDRESS INTO THE LOW SIDE OF X REGISTER
LDI XH, HIGH(STARTADDS) ;LOADING THE STARTADDS MEMORY ADDRESS INTO THE HIGH SIDE OF X REGISTER
LDI YL, LOW(DIVISABLE) ;LOADING THE DIVISABLE MEMORY ADDRESS INTO THE LOW SIDE OF Y REGISTER
LDI YH, HIGH(DIVISABLE) ;LOADING THE DIVISABLE MEMORY ADDRESS INTO THE HIGH SIDE OF Y REGISTER
LDI ZL, LOW(NONDIVISABLE) ;LOADING THE NONDIVISABLE MEMORY ADDRESS INTO THE LOW SIDE OF THE Z REGISTER
LDI ZH, HIGH(NONDIVISABLE) ;LOADING THE NONDIVISABLE MEMORY ADDRESS INTO THE HIGH SIDE OF THE Z REGISTER
LDI R25, DIVIS ;LOADING THE DIVIS VALUE (0x05) INTO R25
CALL FILL_VALS ;CALLING THE FUNCTION FILL_VALS
POST: ;LABEL POST: RETURN POINT OF BADDLY_DIVIS AND GOODLY_DIVIS
LD R24,X+ ;LOAD THE VALUE STORED AT LOCATION POINTED TO BY X, INTO REGISTER R24, THE INCREMENT POINTER VALUE
CPI R24, 0x00 ;COMPARE THIS VALUE TO ZERO IN ORDER TO DETERMIN IF THE END OF THE SET HAS BEEN REACHED
BREQ NEXT ;IF (R24 == 0) BRANCH TO THE LOCATION "NEXT;" MEANING THE ORGINIZATION PHAZE IS OVER
MOV R23,R24 ;COPY R24 INTO R23 IN ORDER TO PRESERVE THE VALUE HELD BY R24 WHEN CALLING IS_DIVIS MACRO
IS_DIVIS R23 ;CALLING IS_DIVIS MACRO
CPI R16, 0x00 ;COMPARE R16 (BOOLEAN 0/1) WITH ZERO
BREQ BADDLY_DIVIS ;BRANCH IF R16 RETURNS AS ZERO (VALUE IN R24 IS NOT DIVISABLE BY 5)
109 %
Processor Status
Name
Value
Program Counter 0x00000001
Stack Pointer 0x08FF
X Register 0x0000
Y Register 0x0000
Z Register 0x0000
Status Register I T H S V N Z C
Cycle Counter 1
Frequency 16.000 MHz
Stop Watch 0.06 µs
Registers
R00 0x00
R01 0x00
R02 0x00
R03 0x00
R04 0x00
R05 0x00
R06 0x00
R07 0x00
R08 0x00
R09 0x00
R10 0x00
R11 0x00
R12 0x00
R13 0x00
R14 0x00
R15 0x00
R16 0x00
R17 0x00
R18 0x00
Terminal Window I/O Processor Status
Watch 1
Name
Value
Type
R16 0x00 byte{reg
R23 0x00 byte{reg
R25 0x00 byte{reg
R16 0x00 byte{reg
R17 0x00 byte{reg
R18 0x00 byte{reg
R19 0x00 byte{reg
Autos Locals Watch 1 Watch 2
Memory 4
Memory: data IRAM
Address: 0x0200,data
data 0x0200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
data 0x021D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
data 0x023A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
data 0x0257 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
data 0x0274 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
data 0x0291 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
data 0x02AE 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
data 0x02CB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
data 0x02E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
Call Stack Breakpoints Memory 4

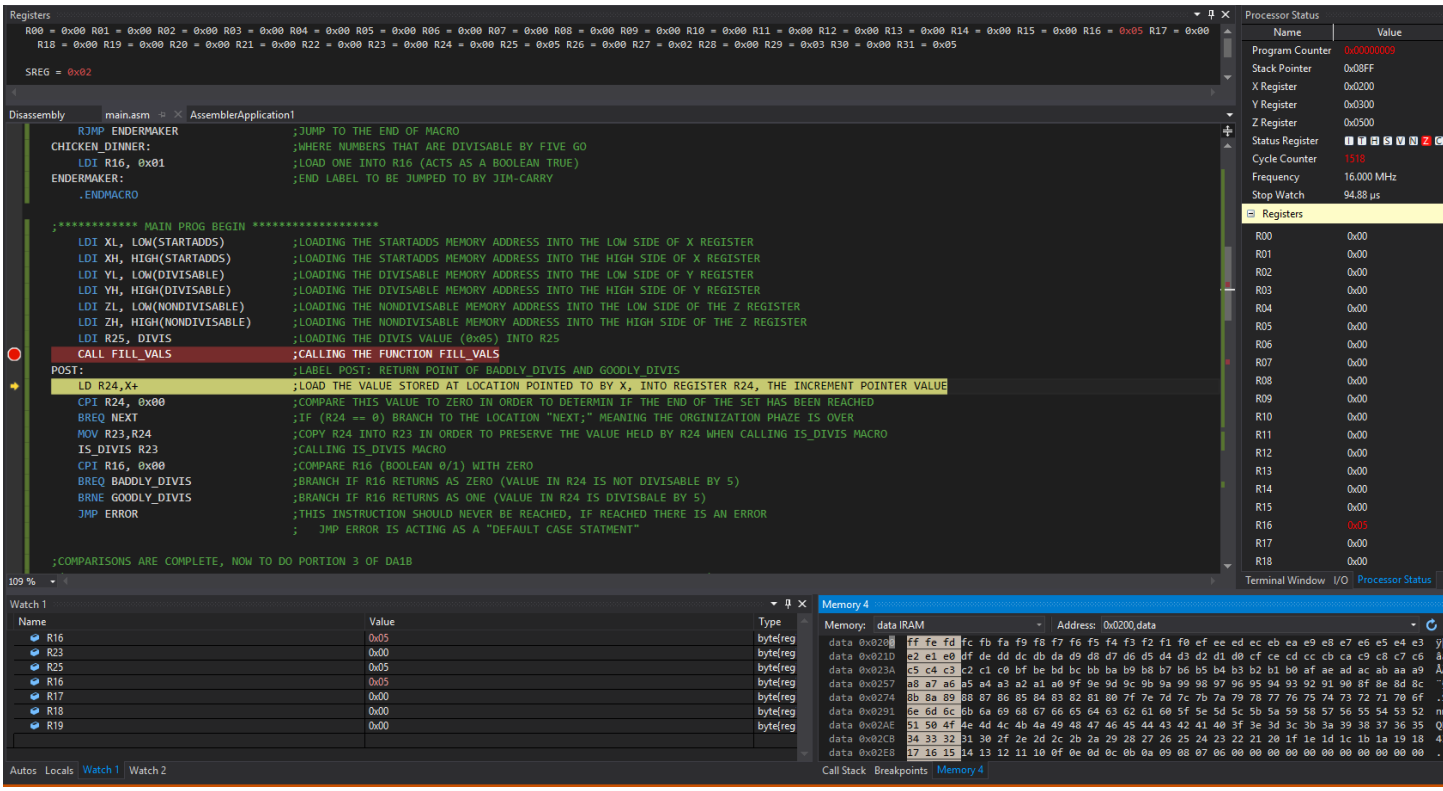
Pre 1


Registers
R00 = 0x00 R01 = 0x00 R02 = 0x00 R03 = 0x00 R04 = 0x00 R05 = 0x00 R06 = 0x00 R07 = 0x00 R08 = 0x00 R09 = 0x00 R10 = 0x00 R11 = 0x00 R12 = 0x00 R13 = 0x00 R14 = 0x00 R15 = 0x00 R16 = 0x00 R17 = 0x00
R18 = 0x00 R19 = 0x00 R20 = 0x00 R21 = 0x00 R22 = 0x00 R23 = 0x00 R24 = 0x00 R25 = 0x00 R26 = 0x00 R27 = 0x00 R28 = 0x00 R29 = 0x00 R30 = 0x00 R31 = 0x00
SREG = 0x00
Disassembly
main.asm
AssemblerApplication1
SUBI R16,1 ;SUBTRACT 1 FROM THE DIVIDEND IN R16
BREQ JIM_CARRY ;IF EQUAL TO ZERO, IT'S A LOOSAHER (LOSER) BREAK TO JIM_CARRY
RJMP DIVISABATOR ;IF FAILD ALL THESE CHECKS, JUMP BACK UP TO TOP OF LOOP
JIM_CARRY: ;WHERE NON-DIVISABLE BY FIVE NUMBERS GO
LDI R16, 0x00 ;LOAD ZERO INTO R16 (ACTS AS A BOOLEAN FALSE)
RJMP ENDERMAKER ;JUMP TO THE END OF MACRO
CHICKEN_DINNER: ;WHERE NUMBERS THAT ARE DIVISABLE BY FIVE GO
LDI R16, 0x01 ;LOAD ONE INTO R16 (ACTS AS A BOOLEAN TRUE)
ENDERMAKER: ;END LABEL TO BE JUMPED TO BY JIM-CARRY
.ENDMACRO
;************ MAIN PROG BEGIN ******************
LDI XL, LOW(STARTADDS) ;LOADING THE STARTADDS MEMORY ADDRESS INTO THE LOW SIDE OF X REGISTER
LDI XH, HIGH(STARTADDS) ;LOADING THE STARTADDS MEMORY ADDRESS INTO THE HIGH SIDE OF X REGISTER
LDI YL, LOW(DIVISABLE) ;LOADING THE DIVISABLE MEMORY ADDRESS INTO THE LOW SIDE OF Y REGISTER
LDI YH, HIGH(DIVISABLE) ;LOADING THE DIVISABLE MEMORY ADDRESS INTO THE HIGH SIDE OF Y REGISTER
LDI ZL, LOW(NONDIVISABLE) ;LOADING THE NONDIVISABLE MEMORY ADDRESS INTO THE LOW SIDE OF THE Z REGISTER
LDI ZH, HIGH(NONDIVISABLE) ;LOADING THE NONDIVISABLE MEMORY ADDRESS INTO THE HIGH SIDE OF THE Z REGISTER
LDI R25, DIVIS ;LOADING THE DIVIS VALUE (0x05) INTO R25
CALL FILL_VALS ;CALLING THE FUNCTION FILL_VALS
POST: ;LABEL POST: RETURN POINT OF BADDLY_DIVIS AND GOODLY_DIVIS
LD R24,X+ ;LOAD THE VALUE STORED AT LOCATION POINTED TO BY X, INTO REGISTER R24, THE INCREMENT POINTER VALUE
CPI R24, 0x00 ;COMPARE THIS VALUE TO ZERO IN ORDER TO DETERMIN IF THE END OF THE SET HAS BEEN REACHED
BREQ NEXT ;IF (R24 == 0) BRANCH TO THE LOCATION "NEXT;" MEANING THE ORGINIZATION PHAZE IS OVER
MOV R23,R24 ;COPY R24 INTO R23 IN ORDER TO PRESERVE THE VALUE HELD BY R24 WHEN CALLING IS_DIVIS MACRO
IS_DIVIS R23 ;CALLING IS_DIVIS MACRO
CPI R16, 0x00 ;COMPARE R16 (BOOLEAN 0/1) WITH ZERO
BREQ BADDLY_DIVIS ;BRANCH IF R16 RETURNS AS ZERO (VALUE IN R24 IS NOT DIVISABLE BY 5)
109 %
Processor Status
Name Value
Program Counter 0x00000001
Stack Pointer 0x08FF
X Register 0x0000
Y Register 0x0000
Z Register 0x0000
Status Register I T H S V N Z C
Cycle Counter 1
Frequency 16.000 MHz
Stop Watch 0.06 µs
Registers
R00 0x00
R01 0x00
R02 0x00
R03 0x00
R04 0x00
R05 0x00
R06 0x00
R07 0x00
R08 0x00
R09 0x00
R10 0x00
R11 0x00
R12 0x00
R13 0x00
R14 0x00
R15 0x00
R16 0x00
R17 0x00
R18 0x00
Terminal Window I/O Processor Status
Watch 1
Name Value Type
R16 0x00 byte{reg
R23 0x00 byte{reg
R25 0x00 byte{reg
R16 0x00 byte{reg
R17 0x00 byte{reg
R18 0x00 byte{reg
R19 0x00 byte{reg
Memory 4
Memory: data IRAM
Address: 0x0200,data
data 0x0200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x021D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x023A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x0257 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x0274 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x0291 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x02AE 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x02CB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x02E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

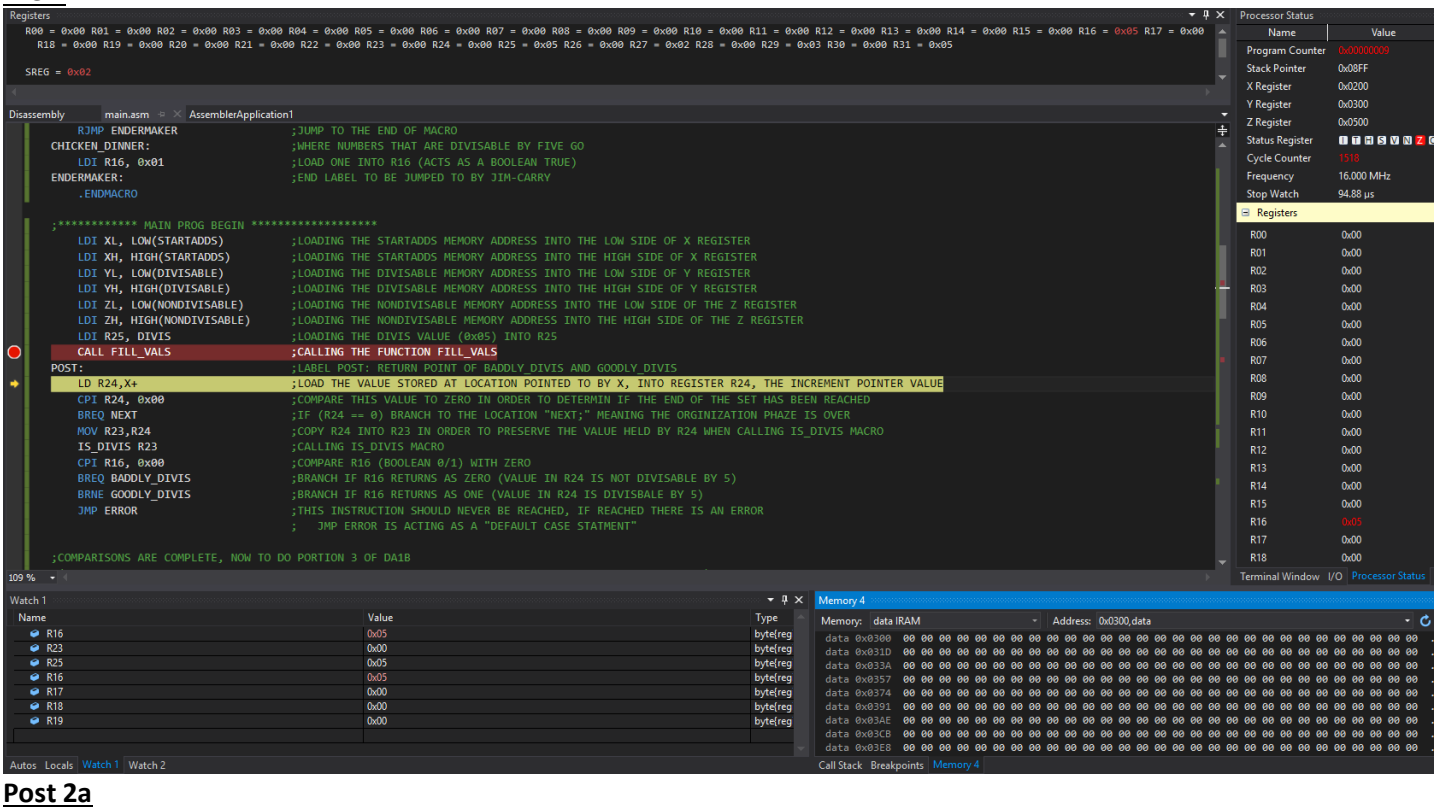
Pre 1


Registers
R00 = 0x00 R01 = 0x00 R02 = 0x00 R03 = 0x00 R04 = 0x00 R05 = 0x00 R06 = 0x00 R07 = 0x00 R08 = 0x00 R09 = 0x00 R10 = 0x00 R11 = 0x00 R12 = 0x00 R13 = 0x00 R14 = 0x00 R15 = 0x00 R16 = 0x00 R17 = 0x00
R18 = 0x00 R19 = 0x00 R20 = 0x00 R21 = 0x00 R22 = 0x00 R23 = 0x00 R24 = 0x00 R25 = 0x05 R26 = 0x00 R27 = 0x02 R28 = 0x00 R29 = 0x03 R30 = 0x00 R31 = 0x05
SREG = 0x00
Disassembly
main.asm
AssemblerApplication1
SUBI R16,1 ;SUBTRACT 1 FROM THE DIVIDEND IN R16
BREQ JIM_CARRY ;IF EQUAL TO ZERO, IT'S A LOOSAHER (LOSER) BREAK TO JIM_CARRY
RJMP DIVISABATOR ;IF FAILD ALL THESE CHECKS, JUMP BACK UP TO TOP OF LOOP
JIM_CARRY: ;WHERE NON-DIVISABLE BY FIVE NUMBERS GO
LDI R16, 0x00 ;LOAD ZERO INTO R16 (ACTS AS A BOOLEAN FALSE)
RJMP ENDERMAKER ;JUMP TO THE END OF MACRO
CHICKEN_DINNER: ;WHERE NUMBERS THAT ARE DIVISABLE BY FIVE GO
LDI R16, 0x01 ;LOAD ONE INTO R16 (ACTS AS A BOOLEAN TRUE)
ENDERMAKER: ;END LABEL TO BE JUMPED TO BY JIM-CARRY
.ENDMACRO
;************ MAIN PROG BEGIN ******************
LDI XL, LOW(STARTADDS) ;LOADING THE STARTADDS MEMORY ADDRESS INTO THE LOW SIDE OF X REGISTER
LDI XH, HIGH(STARTADDS) ;LOADING THE STARTADDS MEMORY ADDRESS INTO THE HIGH SIDE OF X REGISTER
LDI YL, LOW(DIVISABLE) ;LOADING THE DIVISABLE MEMORY ADDRESS INTO THE LOW SIDE OF Y REGISTER
LDI YH, HIGH(DIVISABLE) ;LOADING THE DIVISABLE MEMORY ADDRESS INTO THE HIGH SIDE OF Y REGISTER
LDI ZL, LOW(NONDIVISABLE) ;LOADING THE NONDIVISABLE MEMORY ADDRESS INTO THE LOW SIDE OF THE Z REGISTER
LDI ZH, HIGH(NONDIVISABLE) ;LOADING THE NONDIVISABLE MEMORY ADDRESS INTO THE HIGH SIDE OF THE Z REGISTER
LDI R25, DIVIS ;LOADING THE DIVIS VALUE (0x05) INTO R25
CALL FILL_VALS ;CALLING THE FUNCTION FILL_VALS
POST: ;LABEL POST: RETURN POINT OF BADDLY_DIVIS AND GOODLY_DIVIS
LD R24,X+ ;LOAD THE VALUE STORED AT LOCATION POINTED TO BY X, INTO REGISTER R24, THE INCREMENT POINTER VALUE
CPI R24, 0x00 ;COMPARE THIS VALUE TO ZERO IN ORDER TO DETERMIN IF THE END OF THE SET HAS BEEN REACHED
BREQ NEXT ;IF (R24 == 0) BRANCH TO THE LOCATION "NEXT;" MEANING THE ORGINIZATION PHAZE IS OVER
MOV R23,R24 ;COPY R24 INTO R23 IN ORDER TO PRESERVE THE VALUE HELD BY R24 WHEN CALLING IS_DIVIS MACRO
IS_DIVIS R23 ;CALLING IS_DIVIS MACRO
CPI R16, 0x00 ;COMPARE R16 (BOOLEAN 0/1) WITH ZERO
BREQ BADDLY_DIVIS ;BRANCH IF R16 RETURNS AS ZERO (VALUE IN R24 IS NOT DIVISABLE BY 5)
109 %
Processor Status
Name Value
Program Counter 0x00000007
Stack Pointer 0x08FF
X Register 0x0200
Y Register 0x0300
Z Register 0x0500
Status Register I T H S V N Z C
Cycle Counter 7
Frequency 16.000 MHz
Stop Watch 0.44 µs
Registers
R00 0x00
R01 0x00
R02 0x00
R03 0x00
R04 0x00
R05 0x00
R06 0x00
R07 0x00
R08 0x00
R09 0x00
R10 0x00
R11 0x00
R12 0x00
R13 0x00
R14 0x00
R15 0x00
R16 0x00
R17 0x00
R18 0x00
Terminal Window I/O Processor Status
Watch 1
Name Value Type
R16 0x00 byte{reg
R23 0x00 byte{reg
R25 0x05 byte{reg
R16 0x00 byte{reg
R17 0x00 byte{reg
R18 0x00 byte{reg
R19 0x00 byte{reg
Memory 4
Memory: data IRAM
Address: 0x0200,data
data 0x0200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x021D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x023A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x0257 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x0274 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x0291 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x02AE 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x02CB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x02E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Autos Locals Watch 1 Watch 2
Call Stack Breakpoints Memory 4

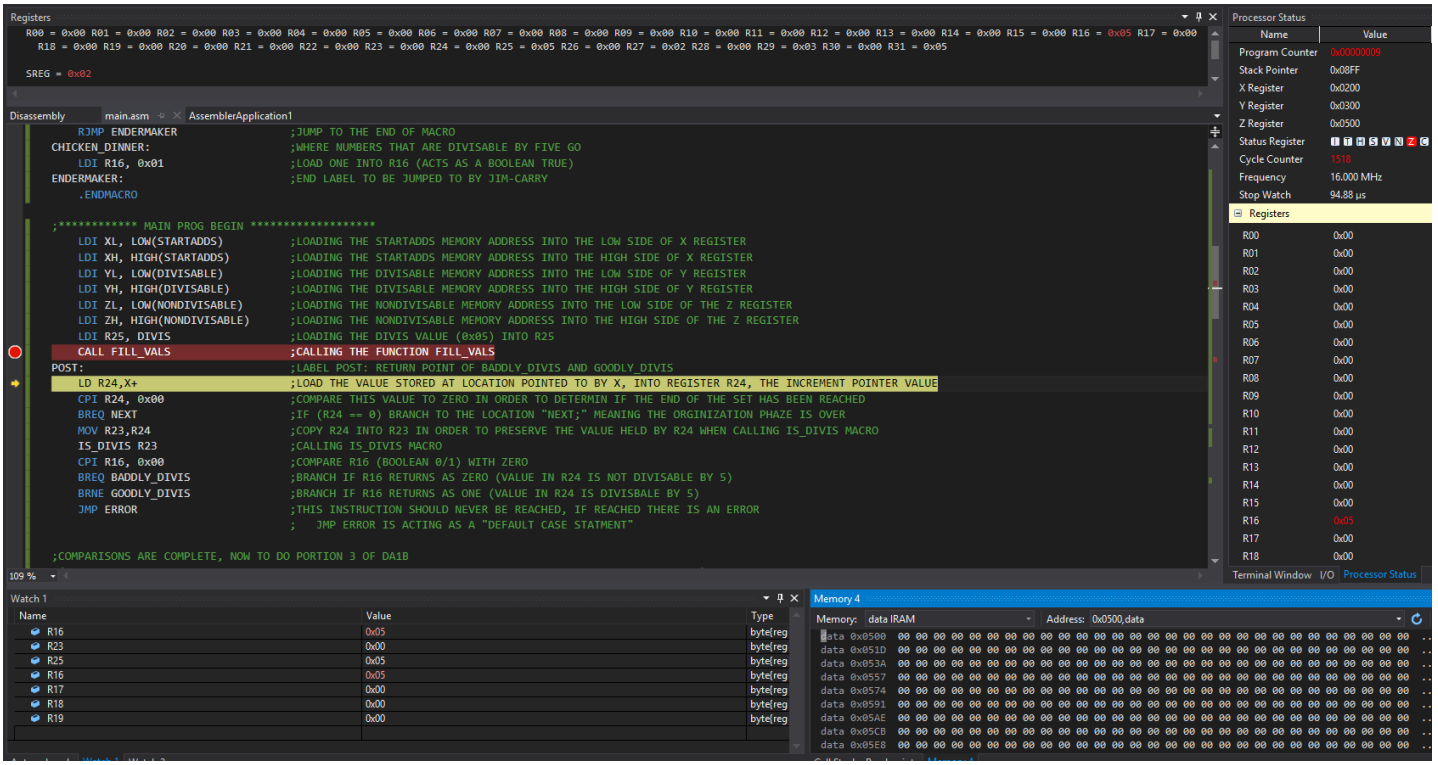
Post 1


Registers
R00 = 0x00 R01 = 0x00 R02 = 0x00 R03 = 0x00 R04 = 0x00 R05 = 0x00 R06 = 0x00 R07 = 0x00 R08 = 0x00 R09 = 0x00 R10 = 0x00 R11 = 0x00 R12 = 0x00 R13 = 0x00 R14 = 0x00 R15 = 0x00 R16 = 0x05 R17 = 0x00
R18 = 0x00 R19 = 0x00 R20 = 0x00 R21 = 0x00 R22 = 0x00 R23 = 0x00 R24 = 0x00 R25 = 0x05 R26 = 0x00 R27 = 0x02 R28 = 0x00 R29 = 0x03 R30 = 0x00 R31 = 0x05
SREG = 0x02
Disassembly
main.asm
AssemblerApplication1
RJMP ENDERMAKER ;JUMP TO THE END OF MACRO
CHICKEN_DINNER: ;WHERE NUMBERS THAT ARE DIVISABLE BY FIVE GO
LDI R16, 0x01 ;LOAD ONE INTO R16 (ACTS AS A BOOLEAN TRUE)
ENDERMAKER: ;END LABEL TO BE JUMPED TO BY JIM-CARRY
.ENDMACRO
;************ MAIN PROG BEGIN ******************
LDI XL, LOW(STARTADDS) ;LOADING THE STARTADDS MEMORY ADDRESS INTO THE LOW SIDE OF X REGISTER
LDI XH, HIGH(STARTADDS) ;LOADING THE STARTADDS MEMORY ADDRESS INTO THE HIGH SIDE OF X REGISTER
LDI YL, LOW(DIVISABLE) ;LOADING THE DIVISABLE MEMORY ADDRESS INTO THE LOW SIDE OF Y REGISTER
LDI YH, HIGH(DIVISABLE) ;LOADING THE DIVISABLE MEMORY ADDRESS INTO THE HIGH SIDE OF Y REGISTER
LDI ZL, LOW(NONDIVISABLE) ;LOADING THE NONDIVISABLE MEMORY ADDRESS INTO THE LOW SIDE OF THE Z REGISTER
LDI ZH, HIGH(NONDIVISABLE) ;LOADING THE NONDIVISABLE MEMORY ADDRESS INTO THE HIGH SIDE OF THE Z REGISTER
LDI R25, DIVIS ;LOADING THE DIVIS VALUE (0x05) INTO R25
CALL FILL_VALS ;CALLING THE FUNCTION FILL_VALS
POST: ;LABEL POST: RETURN POINT OF BADDLY_DIVIS AND GOODLY_DIVIS
LD R24,X+ ;LOAD THE VALUE STORED AT LOCATION POINTED TO BY X, INTO REGISTER R24, THE INCREMENT POINTER VALUE
CPI R24, 0x00 ;COMPARE THIS VALUE TO ZERO IN ORDER TO DETERMIN IF THE END OF THE SET HAS BEEN REACHED
BREQ NEXT ;IF (R24 == 0) BRANCH TO THE LOCATION "NEXT;" MEANING THE ORGINIZATION PHAZE IS OVER
MOV R23,R24 ;COPY R24 INTO R23 IN ORDER TO PRESERVE THE VALUE HELD BY R24 WHEN CALLING IS_DIVIS MACRO
IS_DIVIS R23 ;CALLING IS_DIVIS MACRO
CPI R16, 0x00 ;COMPARE R16 (BOOLEAN 0/1) WITH ZERO
BREQ BADDLY_DIVIS ;BRANCH IF R16 RETURNS AS ZERO (VALUE IN R24 IS NOT DIVISABLE BY 5)
BRNE GOODLY_DIVIS ;BRANCH IF R16 RETURNS AS ONE (VALUE IN R24 IS DIVISBALE BY 5)
JMP ERROR ;THIS INSTRUCTION SHOULD NEVER BE REACHED, IF REACHED THERE IS AN ERROR
; JMP ERROR IS ACTING AS A "DEFAULT CASE STATMENT"
;COMPARISONS ARE COMPLETE, NOW TO DO PORTION 3 OF DA1B
109 %
Processor Status
Name Value
Program Counter 0x00000009
Stack Pointer 0x08FF
X Register 0x0200
Y Register 0x0300
Z Register 0x0500
Status Register I T H S V N Z C
Cycle Counter 1518
Frequency 16.000 MHz
Stop Watch 94.88 µs
Registers
R00 0x00
R01 0x00
R02 0x00
R03 0x00
R04 0x00
R05 0x00
R06 0x00
R07 0x00
R08 0x00
R09 0x00
R10 0x00
R11 0x00
R12 0x00
R13 0x00
R14 0x00
R15 0x00
R16 0x05
R17 0x00
R18 0x00
Terminal Window I/O Processor Status
Watch 1
Name Value Type
R16 0x05 byte{reg
R23 0x00 byte{reg
R25 0x05 byte{reg
R16 0x05 byte{reg
R17 0x00 byte{reg
R18 0x00 byte{reg
R19 0x00 byte{reg
Autos Locals Watch 1 Watch 2
Memory 4
Memory: data IRAM
Address: 0x0200,data
data 0x0200 ff fe fd fc fb fa f9 f8 f7 f6 f5 f4 f3 f2 f1 f0 ef ee ed ec eb ea e9 e8 e7 e6 e5 e4 e3
data 0x021D e2 e1 e0 df de dd dc db da d9 d8 d7 d6 d5 d4 d3 d2 d1 d0 cf ce cd cc cb ca c9 c8 c7 c6
data 0x023A c5 c4 c3 c2 c1 c0 bf be bd bc bb ba b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 af ae ad ac ab aa a9
data 0x0257 a8 a7 a6 a5 a4 a3 a2 a1 a0 9f 9e 9d 9c 9b 9a 99 98 97 96 95 94 93 92 91 90 8f 8e 8d 8c
data 0x0274 8b 8a 89 88 87 86 85 84 83 82 81 80 7f 7e 7d 7c 7b 7a 79 78 77 76 75 74 73 72 71 70 6f
data 0x0291 6e 6d 6c 6b 6a 69 68 67 66 65 64 63 62 61 60 5f 5e 5d 5c 5b 5a 59 58 57 56 55 54 53 52
data 0x02AE 51 50 4f 4e 4d 4c 4b 4a 49 48 47 46 45 44 43 42 41 40 3f 3e 3d 3c 3b 3a 39 38 37 36 35
data 0x02CB 34 33 32 31 30 2f 2e 2d 2c 2b 2a 29 28 27 26 25 24 23 22 21 20 1f 1e 1d 1c 1b 1a 19 18
data 0x02E8 17 16 15 14 13 12 11 10 0f 0e 0d 0c 0b 0a 09 08 07 06 00 00 00 00 00 00 00 00 00 00
Call Stack Breakpoints Memory 4

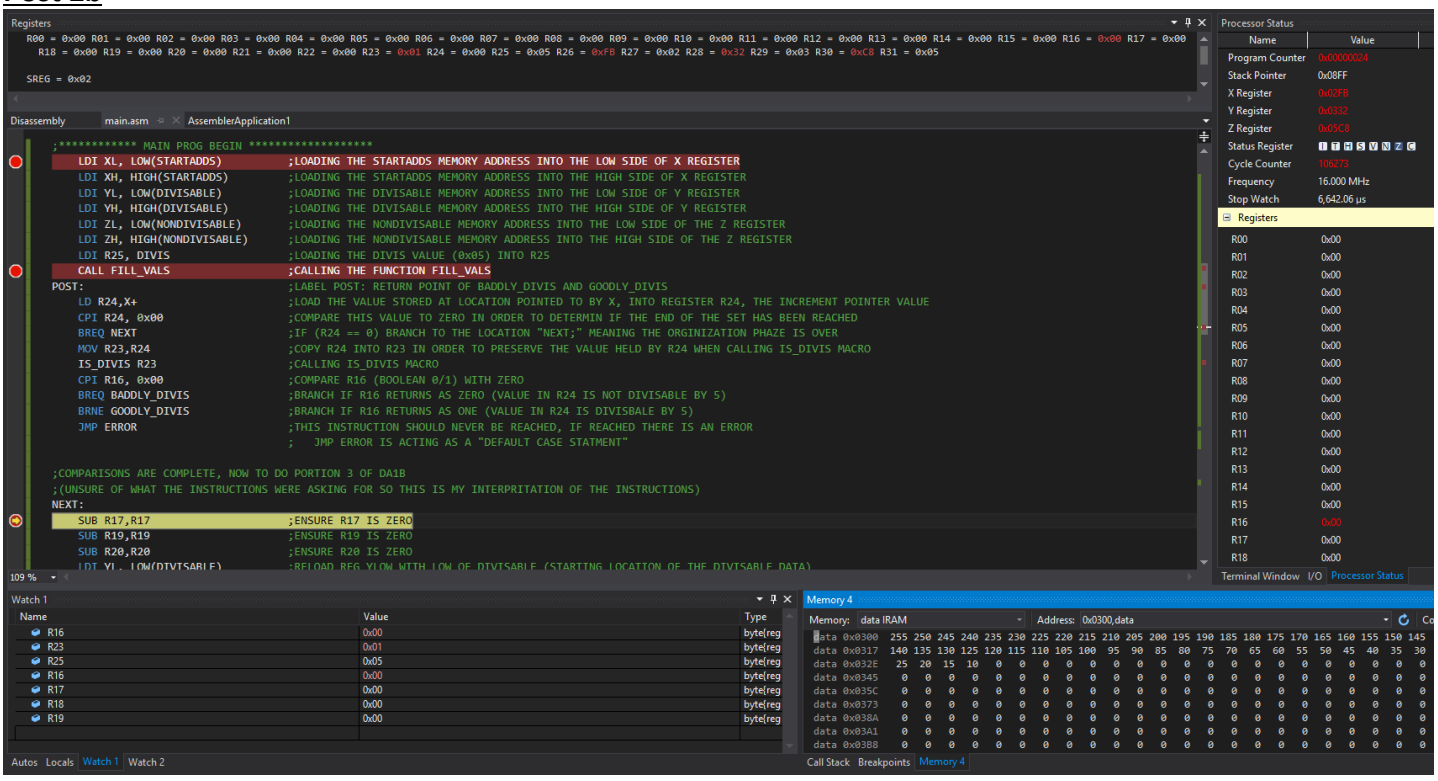
Pre 2


Registers
R00 = 0x00 R01 = 0x00 R02 = 0x00 R03 = 0x00 R04 = 0x00 R05 = 0x00 R06 = 0x00 R07 = 0x00 R08 = 0x00 R09 = 0x00 R10 = 0x00 R11 = 0x00 R12 = 0x00 R13 = 0x00 R14 = 0x00 R15 = 0x00 R16 = 0x05 R17 = 0x00
R18 = 0x00 R19 = 0x00 R20 = 0x00 R21 = 0x00 R22 = 0x00 R23 = 0x00 R24 = 0x00 R25 = 0x05 R26 = 0x00 R27 = 0x02 R28 = 0x00 R29 = 0x03 R30 = 0x00 R31 = 0x05
SREG = 0x02
Disassembly
main.asm
AssemblerApplication1
RJMP ENDERMAKER ;JUMP TO THE END OF MACRO
CHICKEN_DINNER: ;WHERE NUMBERS THAT ARE DIVISABLE BY FIVE GO
LDI R16, 0x01 ;LOAD ONE INTO R16 (ACTS AS A BOOLEAN TRUE)
ENDERMAKER: ;END LABEL TO BE JUMPED TO BY JIM-CARRY
.ENDMACRO
;************ MAIN PROG BEGIN ******************
LDI XL, LOW(STARTADDS) ;LOADING THE STARTADDS MEMORY ADDRESS INTO THE LOW SIDE OF X REGISTER
LDI XH, HIGH(STARTADDS) ;LOADING THE STARTADDS MEMORY ADDRESS INTO THE HIGH SIDE OF X REGISTER
LDI YL, LOW(DIVISABLE) ;LOADING THE DIVISABLE MEMORY ADDRESS INTO THE LOW SIDE OF Y REGISTER
LDI YH, HIGH(DIVISABLE) ;LOADING THE DIVISABLE MEMORY ADDRESS INTO THE HIGH SIDE OF Y REGISTER
LDI ZL, LOW(NONDIVISABLE) ;LOADING THE NONDIVISABLE MEMORY ADDRESS INTO THE LOW SIDE OF THE Z REGISTER
LDI ZH, HIGH(NONDIVISABLE) ;LOADING THE NONDIVISABLE MEMORY ADDRESS INTO THE HIGH SIDE OF THE Z REGISTER
LDI R25, DIVIS ;LOADING THE DIVIS VALUE (0x05) INTO R25
CALL FILL_VALS ;CALLING THE FUNCTION FILL_VALS
POST: ;LABEL POST: RETURN POINT OF BADDLY_DIVIS AND GOODLY_DIVIS
LD R24,X+ ;LOAD THE VALUE STORED AT LOCATION POINTED TO BY X, INTO REGISTER R24, THE INCREMENT POINTER VALUE
CPI R24, 0x00 ;COMPARE THIS VALUE TO ZERO IN ORDER TO DETERMIN IF THE END OF THE SET HAS BEEN REACHED
BREQ NEXT ;IF (R24 == 0) BRANCH TO THE LOCATION "NEXT;" MEANING THE ORGINIZATION PHAZE IS OVER
MOV R23,R24 ;COPY R24 INTO R23 IN ORDER TO PRESERVE THE VALUE HELD BY R24 WHEN CALLING IS_DIVIS MACRO
IS_DIVIS R23 ;CALLING IS_DIVIS MACRO
CPI R16, 0x00 ;COMPARE R16 (BOOLEAN 0/1) WITH ZERO
BREQ BADDLY_DIVIS ;BRANCH IF R16 RETURNS AS ZERO (VALUE IN R24 IS NOT DIVISABLE BY 5)
BRNE GOODLY_DIVIS ;BRANCH IF R16 RETURNS AS ONE (VALUE IN R24 IS DIVISBALE BY 5)
JMP ERROR ;THIS INSTRUCTION SHOULD NEVER BE REACHED, IF REACHED THERE IS AN ERROR
; JMP ERROR IS ACTING AS A "DEFAULT CASE STATMENT"
;COMPARISONS ARE COMPLETE, NOW TO DO PORTION 3 OF DA1B
109 %
Processor Status
Name Value
Program Counter 0x00000009
Stack Pointer 0x08FF
X Register 0x0200
Y Register 0x0300
Z Register 0x0500
Status Register I T H S V N Z C
Cycle Counter 1518
Frequency 16.000 MHz
Stop Watch 94.88 µs
Registers
R00 0x00
R01 0x00
R02 0x00
R03 0x00
R04 0x00
R05 0x00
R06 0x00
R07 0x00
R08 0x00
R09 0x00
R10 0x00
R11 0x00
R12 0x00
R13 0x00
R14 0x00
R15 0x00
R16 0x05
R17 0x00
R18 0x00
Terminal Window I/O Processor Status
Watch 1
Name Value Type
R16 0x05 byte{reg
R23 0x00 byte{reg
R25 0x05 byte{reg
R16 0x05 byte{reg
R17 0x00 byte{reg
R18 0x00 byte{reg
R19 0x00 byte{reg
Autos Locals Watch 1 Watch 2
Memory 4
Memory: data IRAM
Address: 0x0300,data
data 0x0300 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x031D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x033A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x0357 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x0374 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x0391 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x03AE 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x03CB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x03E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Call Stack Breakpoints Memory 4

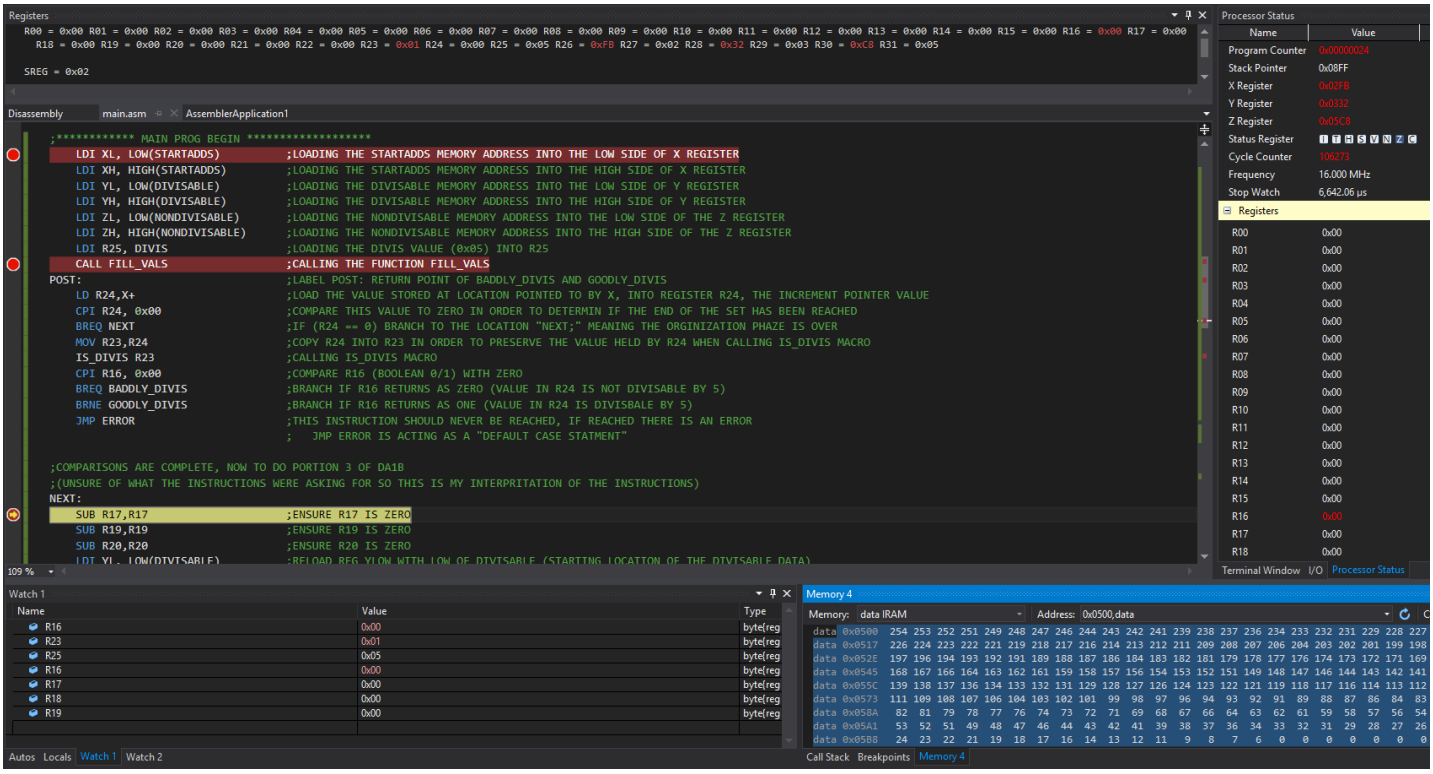
Post 2a



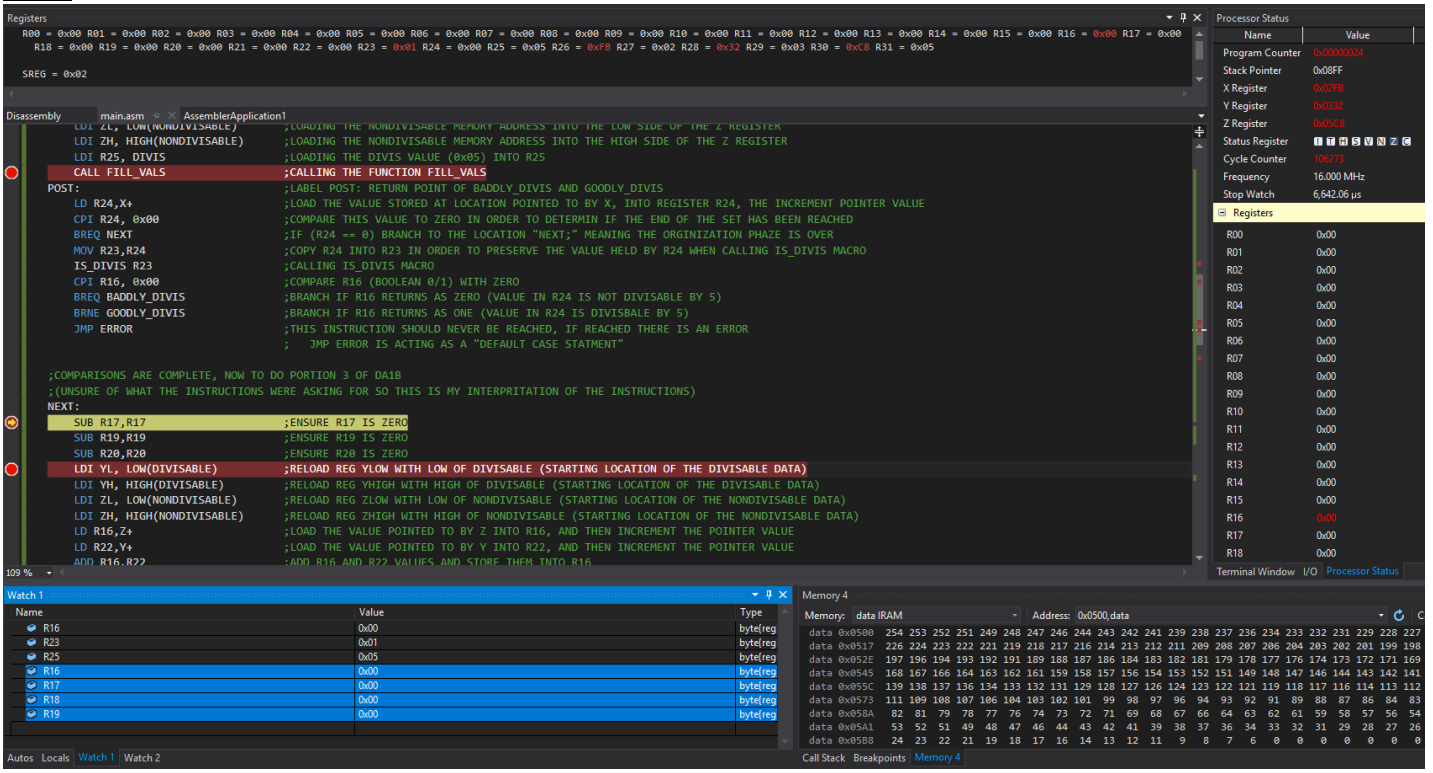
Post 2b



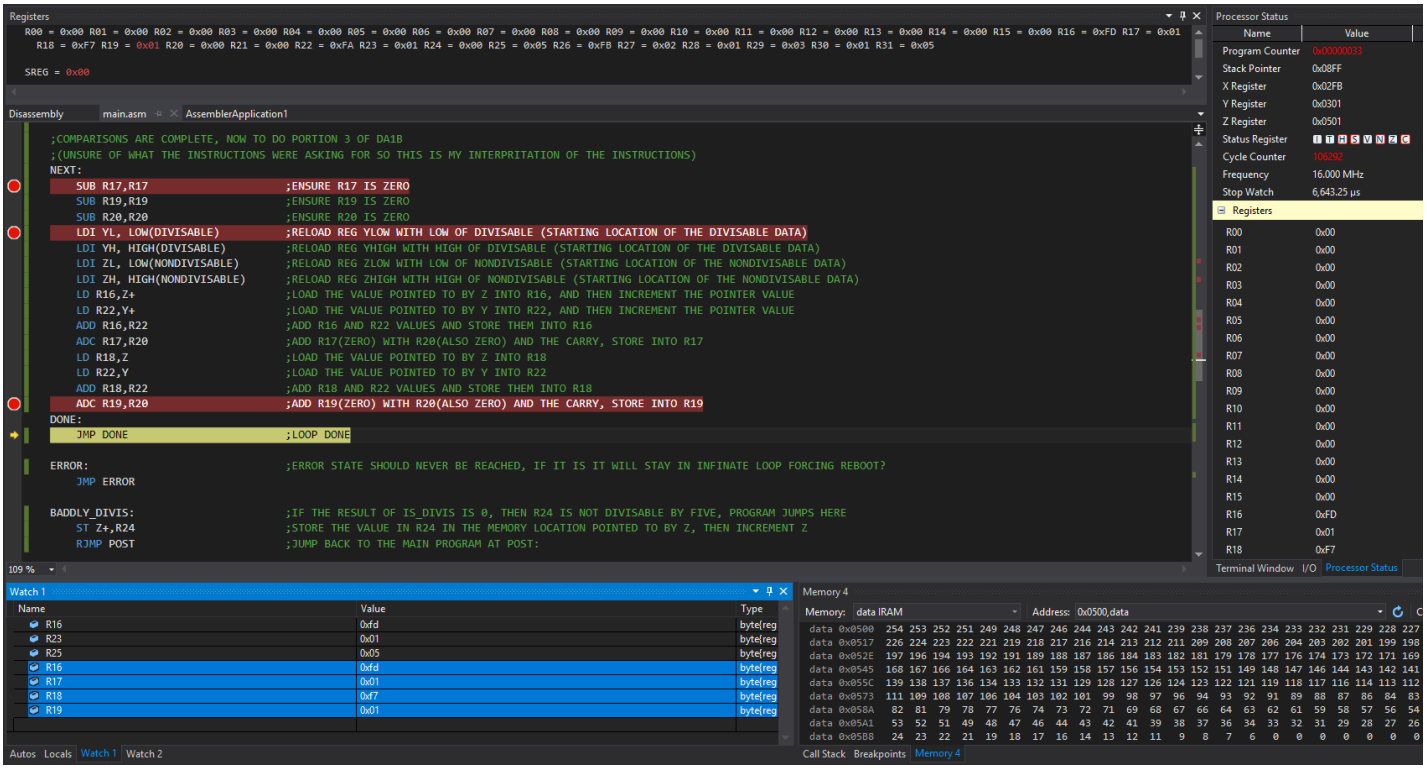
Post 2c


Registers
R00 = 0x00 R01 = 0x00 R02 = 0x00 R03 = 0x00 R04 = 0x00 R05 = 0x00 R06 = 0x00 R07 = 0x00 R08 = 0x00 R09 = 0x00 R10 = 0x00 R11 = 0x00 R12 = 0x00 R13 = 0x00 R14 = 0x00 R15 = 0x00 R16 = 0x00 R17 = 0x00
R18 = 0x00 R19 = 0x00 R20 = 0x00 R21 = 0x00 R22 = 0x00 R23 = 0x01 R24 = 0x00 R25 = 0x05 R26 = 0xFB R27 = 0x02 R28 = 0x32 R29 = 0x03 R30 = 0xC8 R31 = 0x05
SREG = 0x02
Processor Status
Name
Value
Program Counter
0x00000024
Stack Pointer
0x08FF
X Register
0x02FB
Y Register
0x0332
Z Register
0x05C8
Status Register
Cycle Counter
106273
Frequency
16.000 MHz
Stop Watch
6,642.06 µs
Registers
Disassembly
main.asm
AssemblerApplication1
;************ MAIN PROG BEGIN ******************
LDI XL, LOW(STARTADDS) ;LOADING THE STARTADDS MEMORY ADDRESS INTO THE LOW SIDE OF X REGISTER
LDI XH, HIGH(STARTADDS) ;LOADING THE STARTADDS MEMORY ADDRESS INTO THE HIGH SIDE OF X REGISTER
LDI YL, LOW(DIVISABLE) ;LOADING THE DIVISABLE MEMORY ADDRESS INTO THE LOW SIDE OF Y REGISTER
LDI YH, HIGH(DIVISABLE) ;LOADING THE DIVISABLE MEMORY ADDRESS INTO THE HIGH SIDE OF Y REGISTER
LDI ZL, LOW(NONDIVISABLE) ;LOADING THE NONDIVISABLE MEMORY ADDRESS INTO THE LOW SIDE OF THE Z REGISTER
LDI ZH, HIGH(NONDIVISABLE) ;LOADING THE NONDIVISABLE MEMORY ADDRESS INTO THE HIGH SIDE OF THE Z REGISTER
LDI R25, DIVIS ;LOADING THE DIVIS VALUE (0x05) INTO R25
CALL FILL_VALS ;CALLING THE FUNCTION FILL_VALS
POST: ;LABEL POST: RETURN POINT OF BADDLY_DIVIS AND GOODLY_DIVIS
LD R24,X+ ;LOAD THE VALUE STORED AT LOCATION POINTED TO BY X, INTO REGISTER R24, THE INCREMENT POINTER VALUE
CPI R24, 0x00 ;COMPARE THIS VALUE TO ZERO IN ORDER TO DETERMIN IF THE END OF THE SET HAS BEEN REACHED
BREQ NEXT ;IF (R24 == 0) BRANCH TO THE LOCATION "NEXT;" MEANING THE ORGINIZATION PHAZE IS OVER
MOV R23,R24 ;COPY R24 INTO R23 IN ORDER TO PRESERVE THE VALUE HELD BY R24 WHEN CALLING IS_DIVIS MACRO
IS_DIVIS R23 ;CALLING IS_DIVIS MACRO
CPI R16, 0x00 ;COMPARE R16 (BOOLEAN 0/1) WITH ZERO
BREQ BADDLY_DIVIS ;BRANCH IF R16 RETURNS AS ZERO (VALUE IN R24 IS NOT DIVISABLE BY 5)
BRNE GOODLY_DIVIS ;BRANCH IF R16 RETURNS AS ONE (VALUE IN R24 IS DIVISBALE BY 5)
JMP ERROR ;THIS INSTRUCTION SHOULD NEVER BE REACHED, IF REACHED THERE IS AN ERROR
; JMP ERROR IS ACTING AS A "DEFAULT CASE STATMENT"
;COMPARISONS ARE COMPLETE, NOW TO DO PORTION 3 OF DA1B
;(UNSURE OF WHAT THE INSTRUCTIONS WERE ASKING FOR SO THIS IS MY INTERPRITATION OF THE INSTRUCTIONS)
NEXT:
SUB R17,R17 ;ENSURE R17 IS ZERO
SUB R19,R19 ;ENSURE R19 IS ZERO
SUB R20,R20 ;ENSURE R20 IS ZERO
LDI YL, LOW(DIVISABLE) ;RELOAD REG YLOW WITH LOW OF DIVISABLE (STARTING LOCATION OF THE DIVISABLE DATA)
Watch 1
Name
Value
Type
R16 0x00 byte{reg
R23 0x01 byte{reg
R25 0x05 byte{reg
R16 0x00 byte{reg
R17 0x00 byte{reg
R18 0x00 byte{reg
R19 0x00 byte{reg
Memory 4
Memory: data IRAM
Address: 0x0500,data
data 0x0500 254 253 252 251 249 248 247 246 244 243 242 241 239 238 237 236 234 233 232 231 229 228 227
data 0x0517 226 224 223 222 221 219 218 217 216 214 213 212 211 209 208 207 206 204 203 202 201 199 198
data 0x052E 197 196 194 193 192 191 189 188 187 186 184 183 182 181 179 178 177 176 174 173 172 171 169
data 0x0545 168 167 166 164 163 162 161 159 158 157 156 154 153 152 151 149 148 147 146 144 143 142 141
data 0x055C 139 138 137 136 134 133 132 131 129 128 127 126 124 123 122 121 119 118 117 116 114 113 112
data 0x0573 111 109 108 107 106 104 103 102 101 99 98 97 96 94 93 92 91 89 88 87 86 84 83
data 0x058A 82 81 79 78 77 76 74 73 72 71 69 68 67 66 64 63 62 61 59 58 57 56 54
data 0x05A1 53 52 51 49 48 47 46 44 43 42 41 39 38 37 36 34 33 32 31 29 28 27 26
data 0x05B8 24 23 22 21 19 18 17 16 14 13 12 11 9 8 7 6 0 0 0 0 0 0 0
Terminal Window
I/O
Processor Status
Autos
Locals
Watch 1
Watch 2
Call Stack
Breakpoints
Memory 4

Pre 3


Registers
R00 = 0x00 R01 = 0x00 R02 = 0x00 R03 = 0x00 R04 = 0x00 R05 = 0x00 R06 = 0x00 R07 = 0x00 R08 = 0x00 R09 = 0x00 R10 = 0x00 R11 = 0x00 R12 = 0x00 R13 = 0x00 R14 = 0x00 R15 = 0x00 R16 = 0x00 R17 = 0x00
R18 = 0x00 R19 = 0x00 R20 = 0x00 R21 = 0x00 R22 = 0x00 R23 = 0x01 R24 = 0x00 R25 = 0x05 R26 = 0xFB R27 = 0x02 R28 = 0x32 R29 = 0x03 R30 = 0xC8 R31 = 0x05
SREG = 0x02
Processor Status
Name
Value
Program Counter
0x00000024
Stack Pointer
0x08FF
X Register
0x02FB
Y Register
0x0332
Z Register
0x05C8
Status Register
Cycle Counter
106273
Frequency
16.000 MHz
Stop Watch
6,642.06 µs
Registers
Disassembly
main.asm
AssemblerApplication1
LDI ZH, HIGH(NONDIVISABLE) ;LOADING THE NONDIVISABLE MEMORY ADDRESS INTO THE HIGH SIDE OF THE Z REGISTER
LDI R25, DIVIS ;LOADING THE DIVIS VALUE (0x05) INTO R25
CALL FILL_VALS ;CALLING THE FUNCTION FILL_VALS
POST: ;LABEL POST: RETURN POINT OF BADDLY_DIVIS AND GOODLY_DIVIS
LD R24,X+ ;LOAD THE VALUE STORED AT LOCATION POINTED TO BY X, INTO REGISTER R24, THE INCREMENT POINTER VALUE
CPI R24, 0x00 ;COMPARE THIS VALUE TO ZERO IN ORDER TO DETERMIN IF THE END OF THE SET HAS BEEN REACHED
BREQ NEXT ;IF (R24 == 0) BRANCH TO THE LOCATION "NEXT;" MEANING THE ORGINIZATION PHAZE IS OVER
MOV R23,R24 ;COPY R24 INTO R23 IN ORDER TO PRESERVE THE VALUE HELD BY R24 WHEN CALLING IS_DIVIS MACRO
IS_DIVIS R23 ;CALLING IS_DIVIS MACRO
CPI R16, 0x00 ;COMPARE R16 (BOOLEAN 0/1) WITH ZERO
BREQ BADDLY_DIVIS ;BRANCH IF R16 RETURNS AS ZERO (VALUE IN R24 IS NOT DIVISABLE BY 5)
BRNE GOODLY_DIVIS ;BRANCH IF R16 RETURNS AS ONE (VALUE IN R24 IS DIVISBALE BY 5)
JMP ERROR ;THIS INSTRUCTION SHOULD NEVER BE REACHED, IF REACHED THERE IS AN ERROR
; JMP ERROR IS ACTING AS A "DEFAULT CASE STATMENT"
;COMPARISONS ARE COMPLETE, NOW TO DO PORTION 3 OF DA1B
;(UNSURE OF WHAT THE INSTRUCTIONS WERE ASKING FOR SO THIS IS MY INTERPRITATION OF THE INSTRUCTIONS)
NEXT:
SUB R17,R17 ;ENSURE R17 IS ZERO
SUB R19,R19 ;ENSURE R19 IS ZERO
SUB R20,R20 ;ENSURE R20 IS ZERO
LDI YL, LOW(DIVISABLE) ;RELOAD REG YLOW WITH LOW OF DIVISABLE (STARTING LOCATION OF THE DIVISABLE DATA)
LDI YH, HIGH(DIVISABLE) ;RELOAD REG YHIGH WITH HIGH OF DIVISABLE (STARTING LOCATION OF THE DIVISABLE DATA)
LDI ZL, LOW(NONDIVISABLE) ;RELOAD REG ZLOW WITH LOW OF NONDIVISABLE (STARTING LOCATION OF THE NONDIVISABLE DATA)
LDI ZH, HIGH(NONDIVISABLE) ;RELOAD REG ZHIGH WITH HIGH OF NONDIVISABLE (STARTING LOCATION OF THE NONDIVISABLE DATA)
LD R16,Z+ ;LOAD THE VALUE POINTED TO BY Z INTO R16, AND THEN INCREMENT THE POINTER VALUE
LD R22,Y+ ;LOAD THE VALUE POINTED TO BY Y INTO R22, AND THEN INCREMENT THE POINTER VALUE
Watch 1
Name
Value
Type
R16 0x00 byte{reg
R23 0x01 byte{reg
R25 0x05 byte{reg
R16 0x00 byte{reg
R17 0x00 byte{reg
R18 0x00 byte{reg
R19 0x00 byte{reg
Memory 4
Memory: data IRAM
Address: 0x0500,data
data 0x0500 254 253 252 251 249 248 247 246 244 243 242 241 239 238 237 236 234 233 232 231 229 228 227
data 0x0517 226 224 223 222 221 219 218 217 216 214 213 212 211 209 208 207 206 204 203 202 201 199 198
data 0x052E 197 196 194 193 192 191 189 188 187 186 184 183 182 181 179 178 177 176 174 173 172 171 169
data 0x0545 168 167 166 164 163 162 161 159 158 157 156 154 153 152 151 149 148 147 146 144 143 142 141
data 0x055C 139 138 137 136 134 133 132 131 129 128 127 126 124 123 122 121 119 118 117 116 114 113 112
data 0x0573 111 109 108 107 106 104 103 102 101 99 98 97 96 94 93 92 91 89 88 87 86 84 83
data 0x058A 82 81 79 78 77 76 74 73 72 71 69 68 67 66 64 63 62 61 59 58 57 56 54
data 0x05A1 53 52 51 49 48 47 46 44 43 42 41 39 38 37 36 34 33 32 31 29 28 27 26
data 0x05B8 24 23 22 21 19 18 17 16 14 13 12 11 9 8 7 6 0 0 0 0 0 0 0
Terminal Window
I/O
Processor Status
Autos
Locals
Watch 1
Watch 2
Call Stack
Breakpoints
Memory 4

Post 3



Part 4

C++ code and output

```
#include <iostream>

using namespace std;

int main(void)
{
   int intList[250];
int divisable[30];
int notDivisable[250];
int max = 255;
int inc = (max-5);
int divis = 0;
int notdivis = 0;

for(int i = 0; i<(inc); i++)
{
int temp = max - i;
intList[i] = temp;
}

for(int ii = 0; ii<(inc); ii++)
{
int temp = intList[ii]%5;
```

```
if(temp != 0){
notDivisable[notdivis]=intList[ii];
notdivis++;
}else{
divisable[divis]=intList[ii];
divis++;
}
}

cout << "quantity of numbers divisible by five in total prog" << endl;
cout << divis << endl;
cout << "the numbers divisible by 5 are " << endl;
for(int i = 0; i < divis; i++)
{
cout << divisable[i] << ' ' ;
}
cout << endl;
cout << "quantity of numbers not divisible by five in total prog" << endl;
cout << notdivis << endl;
cout << "those numbers are" << endl;
for(int i = 0; i < notdivis; i++)
{
cout << notDivisable[i] << ' ' ;
}
return 0;
}
```

Output part 4

quantity of numbers divisible by five in total prog
50
the numbers divisible by 5 are
255 250 245 240 235 230 225 220 215 210 205 200 195 190 185 180 175 170 165 160 155 150 145 140 135 130 125 120 115 110 105 100 95 90 85 80 75 70 65 60 55 50 45 40 35 30 25 20 15 10
quantity of numbers not divisible by five in total prog
200
those numbers are
254 253 252 251 249 248 247 246 244 243 242 241 239 238 237 236 234 233 232 231 229 228 227 226 224 223 222 221 219 218 217 216 214 213 212 211 209 208 207 206 204 203 202 201 199 198 197 196 194 193 192 191 189 188 187 186 184 183 182 181 179 178 177 176 174 173 172 171 169 168 167 166 164 163 162 161 159 158 157 156 154 153 152 151 149 148 147 146 144 143 142 141 139 138 137 136 134 133 132 131 129 128 127 126 124 123 122 121 119 118 117 116 114 113 112 111 109 108 107 106 104 103 102 101 99 98 97 96 94 93 92 91 89 88 87 86 84 83 82 81 79 78 77 76 74 73 72 71 69 68 67 66 64 63 62 61 59 58 57 56 54 53 52 51 49 48

47 46 44 43 42 41 39 38 37 36 34 33 32 31 29 28 27 26 24 23 22 21 19 18 17 16 14 13 12 11 9 8 7 6

(program exited with code: 0)
Press return to continue

Part 5

Cycle count 106,291 cycles
Program time 6,643.19 us
Or 6.64ms.

6. SCREENSHOT OF EACH DEMO (BOARD SETUP)

N/A

7. VIDEO LINKS OF EACH DEMO

https://youtu.be/_AyfQbos9gU

8. GITHUB LINK OF THIS DA

https://github.com/Dil-bert/Alabaster.git
F067bd66a9c2871e860f43915f827df53d259dfd

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"*This assignment submission is my own, original work*".
NAME OF THE STUDENT