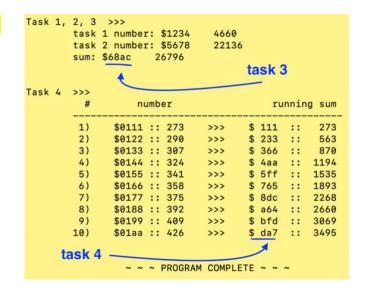
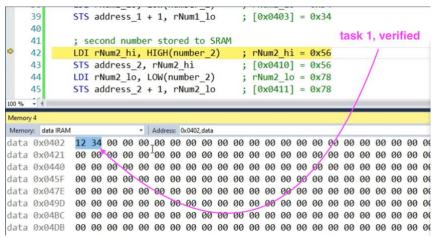
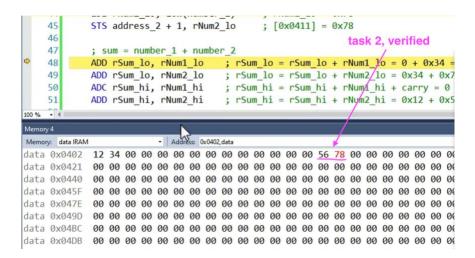
cpe301, Design Assignment 1, David Nakasone



1:: Store a 16-bit number 0x1234 at SRAM location 0x402 Verify the number stored in the location



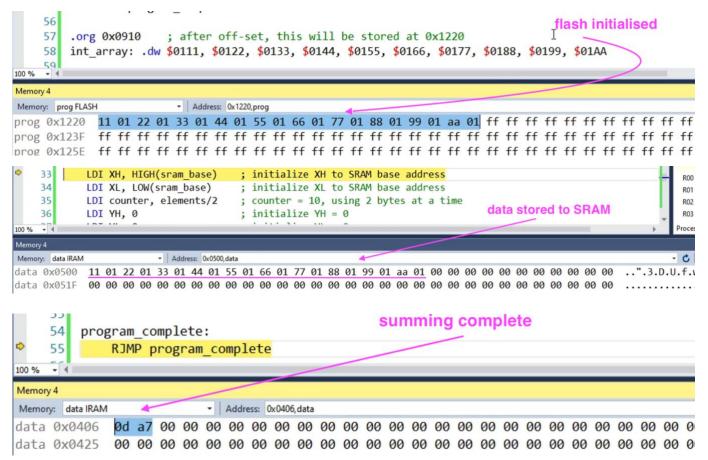
2:: Store a 16-bit number 0x5678 at SRAM location 0x410 Verify the number stored in the location



3:: Sum [0x1234 + 0x5678] and store result in starting EEPROM location Verify the number stored in the location

```
R00 = 0x00 R01 = 0x00 R02 = 0x00 R03 = 0x00 R04 = 0x00 R05 = 0x00 R06 = 0x00 R07
  R10 = 0x00 R11 = 0x00 R12 = 0x00 R13 = 0x00 R14 = 0x00 R15 = 0x00 R16 = 0x00 F
  R20 = 0x12 R21 = 0x34 R22 = 0x56 R23 = 0x78 R24 = 0x68 R25 =
                                                0xAC R26 = 0x01 F
  R30 = 0x00 R31 = 0x00
da1_combined
         main.asm ≠ X
         RJMP program complete
                             loop to stay alive
   63
   64
                                     sum stored in EEPROM
   65
      write to eeprom:
                                 ; (ready to write EEPROM) ? skip next
                  EECR, EEPE
   66
            SBIC
                                 ; not ready to write EEPROM, try agai
   67
            RJMP
                  write to eeprom
                                 ; initialize EEPROM (high) address
   68
            OUT
                  EEARH, XH
                  EEARL ,XL
                                 ; initialize EEPROM (low) address
   69
            OUT
100 %
Memory 4
                   Address: 0x0000,eeprom
Memory: eeprom EEPROM
```

4:: store 10 16-bit numbers stating from 0x0910 a program memory location using code and retrieve them to 0x500 SRAM using X pointer. Sum the 10 numbers and store the sum in SRAM location 0x406



```
; CPE 301, David Nakasone
        ; DA 1, task:: 1, 2, 3
     3
            store a 16-bit number (0x1234) into SRAM location [0x0402]
            store a 16-bit number (0x5678) into SRAM location [0x0410]
     5
        ; add the 2 numbers and store in EEPROM starting location [0x00000]
     6
     7
        .include "m328PBdef.inc"
                                     ; ensure MCU definition
     8
     9
        .cseg
    10
        .org 0
                                       ; start instructions from flash[0]
    11
    12
        .def rTemp = r19
                                 ; r19 == rTemp, a helper register
         def rNum1 hi = r20
                                  ; r20 == rNum1 hi, [15:8] > HIGH(number 1)
Output
                                          - | 🖆 | 🚈 | 🚈 | 😜
Show output from: Build
Done building project "dal_combined.asmproj".
                                             - task 1, 2, 3 code is correct
    ===== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ========
```

```
; CPE 301, David Nakasone
; DA_1, task:: 1, 2, 3
          store a 16-bit number (0x1234) into SRAM location [0x0402]
          store a 16-bit number (0x5678) into SRAM location [0x0410]
          add the 2 numbers and store in EEPROM starting location [0x0000]
.include "m328PBdef.inc" ; ensure MCU definition
.cseg
                ; start instructions from flash[0]
.org 0
.def rTemp = r19
                  ; r19 == rTemp, a helper register
.def rNum1_hi = r20 ; r20 == rNum1_hi, [15:8] > HIGH(number_1)
.def rNum1_lo = r21 ; r21 == rNum1_lo, [7:0] > LOW(number_1)
.def rNum2_hi = r22 ; r22 == rNum2_hi, [15:8] > HIGH(number_2)
.def rNum2_lo = r23 ; r23 == rNum2_lo, [7:0] > LOW(number_2)
.def rSum_hi = r24 ; r24 == rSum_hi [15:8] > HIGH(number_1 + number_2)
.def rSum_lo = r25 ; r25 == rSum_lo [7:0] > LOW(number_1 + number_2)
.equ number_1 = 0x1234 ; value of first number
.equ address_1 = 0x0402 ; SRAM address to store first number
.equ number_2 = 0x5678 ; value of second number
.equ address_2 = 0x0410 ; SRAM address to store second number
          ; initialize SP, using r16
          LDI rTemp, HIGH(RAMEND)
          OUT SPH, rTemp
          LDI rTemp, LOW(RAMEND)
          OUT SPH, rTemp
          ; prepare registers to hold sum
          LDI rSum_hi, 0 ; rSum_hi = 0
          LDI rSum_lo, 0 ; rSum_lo = 0
          ; first number stored to SRAM
           LDI \ rNum1\_hi, HIGH(number\_1) \quad ; \ rNum1\_hi = 0x12 
          STS address_1, rNum1_hi ; [0x0402] = 0x12
          LDI rNum1_lo, LOW(number_1) ; rNum1_lo = 0x34
          STS address_1 + 1, rNum1_lo ; [0x0403] = 0x34
          ; second number stored to SRAM
          LDI rNum2_hi, HIGH(number_2) ; rNum2_hi = 0x56
          STS address_2, rNum2_hi ; [0x0410] = 0x56
          LDI rNum2_lo, LOW(number_2) ; rNum2_lo = 0x78
          STS address_2 + 1, rNum2_lo ; [0x0411] = 0x78
```

```
; sum = number 1 + number 2
          ADD rSum_lo, rNum1_lo ; rSum_lo = rSum_lo + rNum1_lo = 0 + 0x34 = 0x34
          ADD rSum_lo, rNum2_lo ; rSum_lo = rSum_lo + rNum2_lo = 0x34 + 0x78 = 0xAC
          ADC rSum_hi, rNum1_hi ; rSum_hi = rSum_hi + rNum1_hi + carry = 0 + 0x12 + 0 = 0x12
          ADD rSum hi, rNum2 hi ; rSum hi = rSum hi + rNum2 hi = 0x12 + 0x56 = 0x68
          ; write the sum into EEPROM
                          ; *XH = 0
          LDI XL, 0
                          ; *XL = 0
          LDI XH, 0
          MOV rTemp, rSum_hi ; rTemp = rSum_hi
          CALL write_to_eeprom ; write upper byte to EEPROM[0x0000]
                         ; *X = 0x0001, increment to next EEPROM location to write
          MOV rTemp, rSum_lo ; rTemp = rSum_lo
          CALL write_to_eeprom ; write lower byte to EEPROM[0x0001]
program_complete:
          RJMP program_complete ; loop to stay alive
write_to_eeprom:
                                             ; (ready to write EEPROM) ? skip next instruction, wait
                    SBIC
                               EECR, EEPE
                    RJMP
                               write_to_eeprom ; not ready to write EEPROM, try again
                    OUT
                                         EEARH, XH
                                                               ; initialize EEPROM (high) address
                                                               ; initialize EEPROM (low) address
                    OUT
                                         EEARL,XL
                    OUT
                                         EEDR, rTemp
                                                         ; place data in EEPROM data register
                    SBI
                                         EECR, EEMPE
                                                                          ; write data, master check
                                         EECR,EEPE
                                                                          ; write data, final check
                    SBI
                    {\sf CALL}\ {\sf read\_from\_eeprom} \quad \  ; {\sf spin}
                    CBI EECR, EEPE
                                         ; reset writting
                    CBI EECR, EERE
                                         ; reset reading
                    RET
read_from_eeprom:
                    SBIC EECR, EEPE
                                        ; check status of EEPROM
                    RJMP read_from_eeprom ; spins until EEPROM ready
                                                     ; pass address, high, using pointer
                    OUT EEARH,XH
                               EEARL,XL
                    OUT
                                                              ; pass address, low, using pointer
                    SBI
                               EECR,EERE
                                                              ; enable reading
                    IN
                               rTemp,EEDR
                                                                         ; verifies write operation
                    RET
;;;;END
```

```
; cpe301, David Nakasone
               ; DA 1, task:: 4
                       store ten 16-bit numbers starting at program memory/flash [0x0910]
         3
         4
                      retrieve the numbers to SRAM [0x500] using X-pointer
                      sum the 10 numbers and store result to SRAM [0x0406]
                .include <m328PBdef.inc>
         9
                .cseg
        10
               .org 0
        11
        12
                .equ sram_base = $0500
                                                                  ; first number here, others follow, 10 total
                                                                   ; the sum will be stored here
                 equ sram sum = $0406
Output
                                                                             - | 🖆 | 🎍 | 🏝 | 🏖
Show output from: Build
                               memory use summary [bytes]:
n End Code Data
            "ATmega328PB"
                                                                               Size
            Segment Begin
                                                                             32768
            [.cseg] 0x000000 0x001234
[.dseg] 0x000100 0x000100
                                                                                         0.2%
                                                                                2048
[.dseg] 0x000100 0x000100 0 0 0 2048 0.0%
[.eseg] 0x000000 0x000100 0 0 0 1024 0.0%
Assembly complete, 0 errors. 0 warnings
Done executing task "RunAssemblerTask".

Done building target "CoreBuild" in project "dal_combined.asmproj".

Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != ''
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "C:\Users\r
Done building target "Build" in project "dal_combined.asmproj".

Done building project "dal_combined.asmproj".

    task 4 code is correct

 ====== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped =======
```

```
; cpe301, David Nakasone
; DA_1, task:: 4
          store ten 16-bit numbers starting at program memory/flash [0x0910]
           retrieve the numbers to SRAM [0x500] using X-pointer
           sum the 10 numbers and store result to SRAM [0x0406]
.include <m328PBdef.inc>
.cseg
.org 0
.equ sram_base = $0500 ; first number here, others follow, 10 total
.equ sram_sum = $0406 ; the sum will be stored here
.egu elements = 20
                     ; there are 10 numbers in the array, each number uses 2 bytes
.def counter = r16
                    ; loop control variable
.def temp = r17
                    ; temporarily hold a byte
.def sum_hi = r18
                     ; running sum, high
.def sum_lo = r19
                     ; running sum, low
           LDI counter, elements ; initialize the counter to 10
           LDI temp, 0
                                              ; initialize the temporary register
           LDI ZH, HIGH(int_array<<1) ; initialize ZH to flash location
           LDI ZL, LOW(int_array<<1) ; initialize ZL to flash location
           LDI XH, HIGH(sram base)
                                     ; initialize XH to SRAM location
           LDI XL, LOW(sram_base)
                                              ; initialize XL to SRAM location
load sram:
           LPM temp, Z+
                                 ; store contents of Z pointer to temp, then increment Z pointer
           ST X+. temp
                                 ; store temp to flash, then increment the X pointer
           DEC counter ; counter--
           BRNE load_sram ; (20 bytes transfered to SRAM) ? continue : keep loading from flash
           LDI XH, HIGH(sram_base) ; initialize XH to SRAM base address
           LDI XL, LOW(sram_base) ; initialize XL to SRAM base address
           LDI counter, elements/2 ; counter = 10, using 2 bytes at a time
           LDI YH. 0
                             : initialize YH = 0
```

```
LDI YL, 0
                           ; initialize YL = 0
          LDI sum_hi, 0
                              ; initialize sum_hi = 0
          LDI sum_lo, 0
                              ; initialize sum_lo = 0
sum_nums:
          LD YL, X+
          LD YH, X+
          ADD sum_lo, YL ; adding, carray flag could be set
          ADC sum_hi, YH ; adding with carry if carry set previously
          DEC counter ; counter--
          BRNE sum_nums ; (10 number summmed) ? continue : keep summing
          LDI XH, HIGH(sram_sum) ; initialize XH to SRAM sum storage location
          LDI XL, LOW(sram_sum) ; initialize XL to SRAM sum storage location
                          ; store sum_hi to location X, then increment X to store next byte
          ST X+, sum_hi
          ST X, sum_lo
                            ; store sum_lo to location X
program_complete:
          RJMP program_complete
.org 0x0910 ; after off-set, this will be stored at 0x1220
int_array: .dw $0111, $0122, $0133, $0144, $0155, $0166, $0177, $0188, $0199, $01AA
;;;;END
```