1. Write assembly language program for microcontroller ATmega328P to find y=x2 + x + 5, and x is between 0 to 9. x is present in RAM location $200. Place the result in RAM location $201.

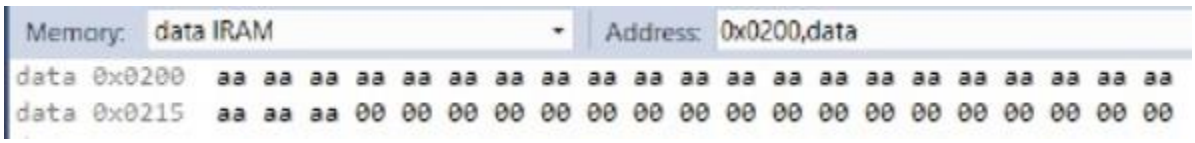
CODE:

```
LDS     R16, 0x200      ; Load x from SRAM[0x200] into R16
MOV     R17, R16        ; Copy x to R17
MUL     R17, R16        ; R0 = x * x (lower byte)
MOV     R18, R0         ; R18 = x^2
ADD     R18, R16        ; R18 = x^2 + x
SUBI    R18, -5         ; R18 = x^2 + x + 5
STS     0x201, R18      ; Store result y to SRAM[0x201]
HERE: RJMP HERE         ; Infinite loop to end program
```

2. Write assembly language program for microcontroller ATmega328P to copy $AA 8-bit value into RAM locations $200 to $ 217.

CODE:

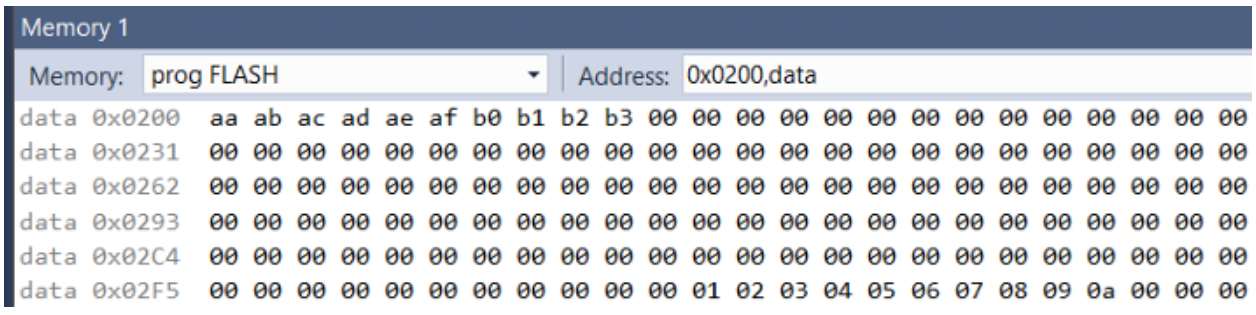
```
LDI     R16, 0xAA          ; Load value $AA into R16
LDI     R30, LOW(0x200)    ; Load low byte of start address into ZL
LDI     R31, HIGH(0x200)   ; Load high byte of start address into ZH
LDI     R17, 24            ; Set loop counter for 24 bytes
AGAIN:
ST      Z+, R16            ; Store $AA at Z, increment Z
DEC     R17                ; Decrement counter
BRNE    AGAIN              ; Repeat if not zero
HERE:
RJMP    HERE               ; Infinite loop (end of program)
```



3. Write assembly language program for microcontroller ATmega328P to exchange the contents of two data blocks of size 10 bytes. First data block starts from data memory address $200 onwards. Second data block starts from data memory address $300 onwards.

CODE:

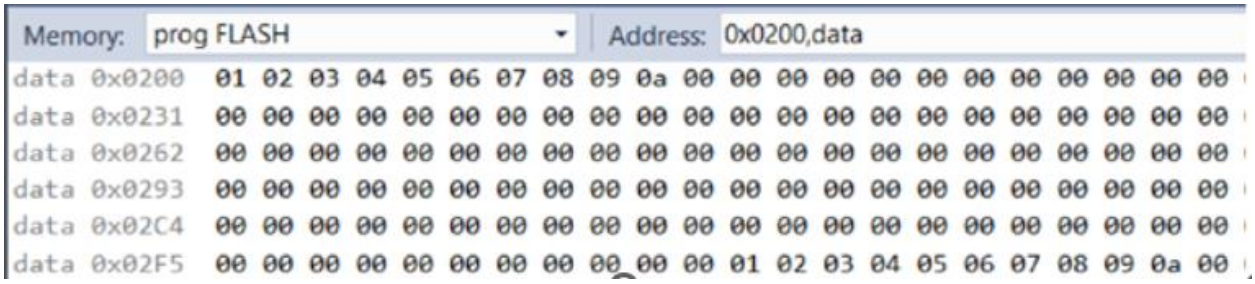
```
LDI R30, LOW(0x200)    ; Load low byte of block 1 start address into ZL

LDI R31, HIGH(0x200)   ; Load high byte of block 1 start address into ZH

LDI R28, LOW(0x300)    ; Load low byte of block 2 start address into YL

LDI R29, HIGH(0x300)   ; Load high byte of block 2 start address into YH

LDI R20, 10            ; Load loop counter (10 bytes to swap)


AGAIN:

LD R16, Z              ; Load byte from block 1 into R16

LD R17, Y              ; Load byte from block 2 into R17

ST Y+, R16             ; Store block 1 byte to block 2, then increment Y

ST Z+, R17             ; Store block 2 byte to block 1, then increment Z

DEC R20                ; Decrement loop counter

BRNE AGAIN             ; Repeat until R20 = 0


HERE:

RJMP HERE              ; Infinite loop (end of program)
```



4. Write assembly language program for microcontroller ATmega328P to transfer the contents of source data block of size 10 bytes. Source data block starts from data memory address $200 onwards. Destination data block starts from data memory address $300 onwards.

CODE:

```
LDI R30, LOW(0x0200)    ; ZL → low byte of source address
LDI R31, HIGH(0x0200)   ; ZH → high byte of source address

LDI R28, LOW(0x0300)    ; YL → low byte of destination address
LDI R29, HIGH(0x0300)   ; YH → high byte of destination address

LDI R20, 10             ; Loop counter = 10 bytes

COPY_LOOP:
LD R16, Z+              ; Load from source and post-increment Z
ST Y+, R16              ; Store to destination and post-increment Y
DEC R20                 ; Decrement counter
BRNE COPY_LOOP          ; Loop until all 10 bytes are copied

HERE:
RJMP HERE               ; Infinite loop (end of program)
```



5. Write assembly language program for microcontroller ATmega328P to convert packed BCD to ASCII codes. Packed BCD is present in RAM location $200. Store the results in RAM locations $201 and $202.

CODE:

```
LDS R16, 0x200        ; Load packed BCD from $200
LDI R18, 0x30         ; ASCII offset ('0') = 0x30

MOV R17, R16          ; Copy to extract lower nibble
ANDI R17, 0x0F        ; Mask upper nibble → get lower BCD digit
ADD R17, R18          ; Convert to ASCII by adding 0x30
STS 0x202, R17        ; Store lower ASCII at $202

SWAP R16              ; Swap nibbles so upper digit is in lower nibble
ANDI R16, 0x0F        ; Mask upper nibble (now in lower 4 bits)
ADD R16, R18          ; Convert to ASCII
STS 0x201, R16        ; Store upper ASCII at $201

HERE:
RJMP HERE             ; Infinite loop
```

```
data 0x0200  45 34 35
```

6. Write assembly language program for microcontroller ATmega328P to find the largest number from data block of 10 unsigned numbers. Data block starts from RAM location $200 onwards. Store the result in RAM location $20A.

```
CODE:
LDI R30, LOW(0x200)     ; Z-pointer low byte → start of data block
LDI R31, HIGH(0x200)    ; Z-pointer high byte
LDI R20, 10             ; Counter = 10 numbers

LD R16, Z+              ; Load first number into R16 → assume it's max

LOOP:
   LD R17, Z+           ; Load next number into R17
   CP R16, R17          ; Compare R16 (max) with R17 (new value)
   BRLO UPDATE_MAX      ; If R17 > R16, update max
   RJMP SKIP_UPDATE     ; Else, skip

UPDATE_MAX:
   MOV R16, R17         ; Update R16 with new max

SKIP_UPDATE:
   DEC R20              ; Decrement counter
   BRNE LOOP            ; Repeat if not 0
STS 0x20A, R16          ; Store largest number in RAM location $20A
HERE:
RJMP HERE               ; Infinite loop (end of program)
```

```
data 0x0200  01 02 03 04 05 06 07 08 09 0a 0a 00 00 00
data 0x020A  0a 00 00 00
```

7. Write assembly language program for microcontroller ATmega328P to add two 16 bit numbers. First 16-bit number present in RAM locations $200 (LB) and $201 (HB). Second 16-bit number present in RAM locations $202 (LB) and $203 (HB). Store the result in RAM locations $ 204, $205 and $206.

```
; Load first 16-bit number
```

```
LDS R16, 0x200      ; Load lower byte of 1st number
LDS R17, 0x201      ; Load upper byte of 1st number

; Load second 16-bit number
LDS R18, 0x202      ; Load lower byte of 2nd number
LDS R19, 0x203      ; Load upper byte of 2nd number

; Perform addition
ADD R16, R18        ; Add lower bytes
ADC R17, R19        ; Add upper bytes + carry
CLR R20             ; Clear R20 (for 3rd byte result)
ADC R20, R20        ; Store carry (if any) from upper byte addition

; Store the result (3 bytes)
STS 0x204, R16      ; Store result lower byte
STS 0x205, R17      ; Store result upper byte
STS 0x206, R20      ; Store carry byte (0 or 1)

HERE:
RJMP HERE           ; Infinite loop to end program
```

```
data 0x0200  34 12 78 56 ac 68
```

8. Write assembly language program for microcontroller ATmega328P to count number of 1s in a given byte which present in RAM location $200. Store the result in RAM location $201.

CODE :

```
LDS R16, 0x200      ; Load byte from RAM location $200
CLR R17             ; R17 = count = 0
```

```
LDI R18, 8          ; R18 = loop counter (8 bits)

LOOP:
   LSL R16          ; Shift left, MSB goes into Carry
   BRCS INC_COUNT   ; If Carry set, bit was 1 → branch to increment count
   RJMP SKIP_INC    ; Else skip increment

INC_COUNT:
   INC R17          ; Increment count

SKIP_INC:
   DEC R18          ; Decrement bit counter
   BRNE LOOP        ; Repeat for all 8 bits

STS 0x201, R17      ; Store result (number of 1s) at $201

HERE:
RJMP HERE           ; End of program (infinite loop)
```

```
data 0x0200  f3 06 00 00 00
```

9. Write assembly language program for microcontroller ATmega328P to add data block of 10 bytes. Data block starts from RAM location $200 onwards. Store the result in RAM location $20A and $20B.

CODE:

```
LDI R30, LOW(0x200)    ; Z pointer → start of data block
LDI R31, HIGH(0x200)
```

```
CLR R16             ; R16 = sum low byte
CLR R17             ; R17 = sum high byte (carry tracker)
LDI R18, 10         ; Counter = 10 bytes

LOOP:
   LD R19, Z+       ; Load data byte into R19
   ADD R16, R19     ; Add to low byte of sum

   BRSH NO_CARRY    ; If no carry, skip increment
   INC R17          ; If carry occurred, increment high byte

NO_CARRY:
   DEC R18          ; Decrement counter
   BRNE LOOP        ; Repeat until all 10 bytes are processed

STS 0x20A, R16      ; Store low byte of result
STS 0x20B, R17      ; Store high byte of result

HERE:
RJMP HERE           ; Infinite loop
```

10. Write assembly language program for microcontroller ATmega328P to convert two unpacked BCD to packed BCD. Unpacked BCD numbers is present in RAM locations $200 and $201. Store the results in RAM locations $203.

```
CODE:
LDS R16, 0x200      ; Load MSD (most significant digit)
LDS R17, 0x201      ; Load LSD (least significant digit)

LSL R16             ; Shift left 1 bit → x2
LSL R16             ; Shift left 1 bit → x4
LSL R16             ; Shift left 1 bit → x8
LSL R16             ; Shift left 1 bit → x16 (i.e., R16 = MSD * 16)

ADD R16, R17        ; Add LSD to get packed BCD

STS 0x203, R16      ; Store packed BCD at $203

HERE:
RJMP HERE           ; Infinite loop

data 0x0200  09 03 00 93
```

11. Write assembly language program for microcontroller ATmega328P to find square of 8-bit number which is present in data memory location $200. Store the result in data memory locations $201 and $202.

```
. CODE:
LDS R16, 0x200      ; Load number from $200 → R16
MOV R17, R16        ; Copy number to R17 (counter)
CLR R18             ; Clear R18 (result low byte)
```

```
CLR R19          ; Clear R19 (result high byte)



LOOP_SQUARE:
ADD R18, R16     ; Add number to result low byte
BRSH NO_CY       ; If no carry, skip
INC R19          ; Else, increment high byte

NO_CY:
DEC R17          ; Decrement counter
BRNE LOOP_SQUARE ; Repeat until R17 = 0

STS 0x201, R19   ; Store high byte of result
STS 0x202, R18   ; Store low byte of result

HERE:
RJMP HERE        ; Infinite loop

data 0x0200  0a 00 64
```

12. Write assembly language program for microcontroller ATmega328P to count even number from a data block of 10 bytes. The data block starts from data memory locations $200 onwards. Store the result at data memory location $20A.

```
CODE:
LDI R30, LOW(0x200)  ; Z-pointer low → start of data block
LDI R31, HIGH(0x200) ; Z-pointer high
LDI R20, 10          ; Counter for 10 bytes
```

```
    CLR R21             ; R21 will store count of even numbers


    LOOP:
    LD R16, Z+          ; Load next byte into R16, post-increment Z
    ANDI R16, 0x01      ; Mask LSB (check if number is even)
    BRNE NOT_EVEN       ; If LSB is 1, it's odd → skip


    INC R21             ; Else, increment even count


    NOT_EVEN:
    DEC R20             ; Decrement loop counter
    BRNE LOOP           ; Repeat until 10 bytes checked


    STS 0x20A, R21      ; Store count of even numbers in $20A


    HERE:
    RJMP HERE           ; Infinite loop

data 0x0200  02 03 04 05 06 07 08 09 0a 0b 05
```

13. Write assembly language program for microcontroller ATmega328P to perform addition of two 16 bit numbers. The first 16-bit number is present in data memory location $200 (LB) and $201 (HB). The second 16-bit number is present in data memory locations $202 (LB) and $203 (HB). Store the result (sum) at data memory locations $204 (LB), $205, and $206.

Same as Q7

14. Write assembly language program for microcontroller ATmega328P to perform subtraction of two 16 bit numbers. The first 16-bit number is present in data memory location $200 (LB) and $201 (HB). The second 16-bit number is present in data memory

locations $202 (LB) and $203 (HB). Store the result at data memory locations $204 (LB) and $205 (HB).

```
CODE:
   ; Load 1st 16-bit number from $200 (LB) and $201 (HB)
   LDS R16, 0x200     ; R16 = Low byte of first number
   LDS R17, 0x201     ; R17 = High byte of first number
   ; Load 2nd 16-bit number from $202 (LB) and $203 (HB)
   LDS R18, 0x202     ; R18 = Low byte of second number
   LDS R19, 0x203     ; R19 = High byte of second number
   ; Perform subtraction: R17:R16 - R19:R18
   SUB R16, R18       ; Subtract low bytes
   SBC R17, R19       ; Subtract high bytes with carry
   ; Store result in $204 (LB) and $205 (HB)
   STS 0x204, R16
   STS 0x205, R17
HERE:
   RJMP HERE;
data 0x0200  11 12 13 14 fe fd
```

15. Write assembly language program for microcontroller ATmega328P to perform 8-bit division using repetitive subtraction. Dividend is present in data memory location $200. Divisor is present in data memory location $201. Store the quotient at data memory location $202 and remainder at data memory location $203.

```
CODE:
   ; Load Dividend and Divisor
   LDS R16, 0x200      ; R16 = Dividend
   LDS R17, 0x201      ; R17 = Divisor
```

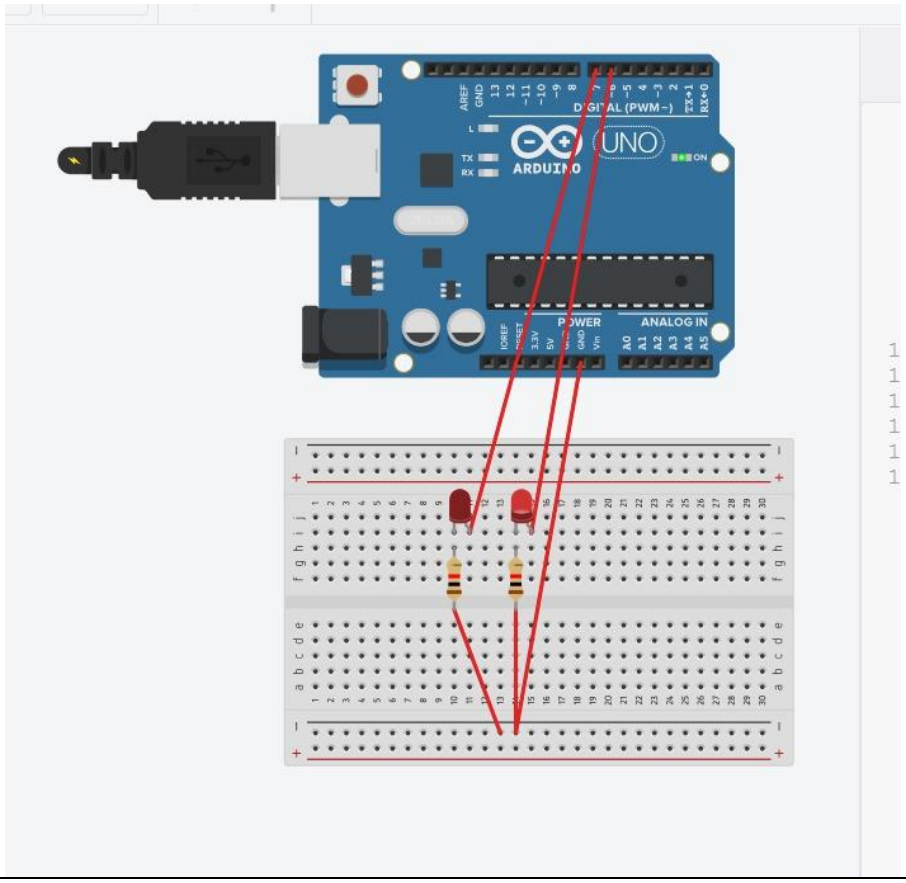
```
    CLR R18          ; R18 = Quotient = 0

DIV_LOOP:
    CP R16, R17      ; Compare Dividend with Divisor
    BRLO DIV_END     ; If Divisor> Dividend, exit loop
    SUB R16, R17     ; Dividend = Dividend - Divisor
    INC R18          ; Increment Quotient
    RJMP DIV_LOOP    ; Repeat

DIV_END:
    STS 0x202, R18   ; Store Quotient
    STS 0x203, R16   ; Store Remainder (what's left of Dividend)
HERE:
    RJMP HERE        ; Infinite loop
data 0x0200  05 02 02 01
```

16. Interface LED to PD7 pin of ATmega328P microcontroller (Arduino Uno board). Write assembly language program to blink the LED at the rate of 1 second. Use crystal frequency 16 MHz.

```
.ORG 0X00
HERE: SBI DDRD, 7
SBI PORTD, 7
CALL DELAY
CBI PORTD, 7
CALL DELAY
RJMP HERE
DELAY:
; Outer loop: 255 x 255 x ~16 cycles = ~1s
LDI R18, $04
LOOP3:
```

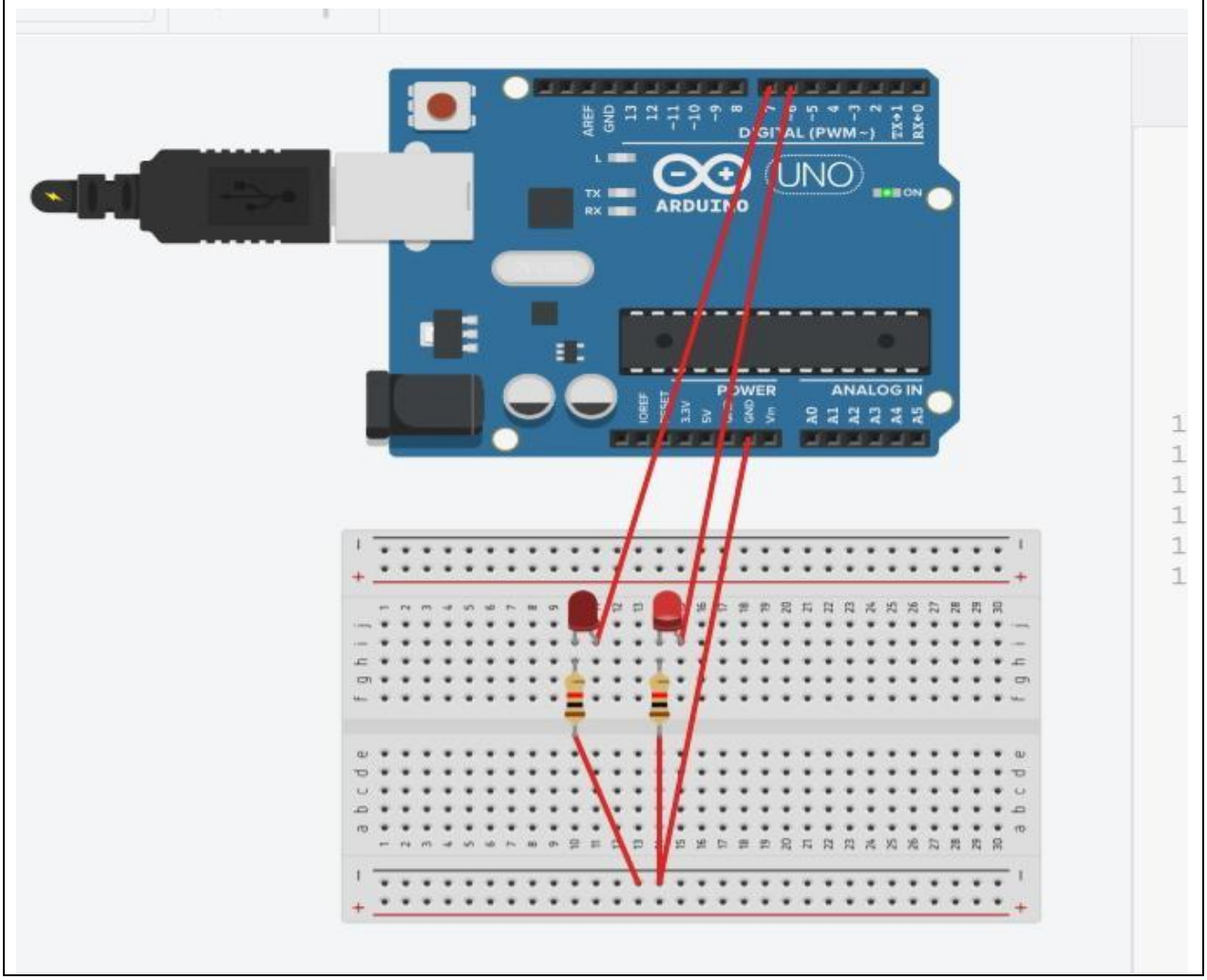
```
LDI R16, $FF
LOOP2:
LDI R17, $FF
LOOP1:
; Delay ~4 cycles per iteration (2 for dec, 2 for brne)
NOP
DEC R17
BRNE LOOP1
DEC R16
BRNE LOOP2
DEC R18
BRNE LOOP3
RET
```

Note: only do for 1 led connection for PD7 connected



17. 16. Interface two LED to PD7 and PD6 pin of ATmega328P microcontroller (Arduino Uno board). Write assembly language program to turn on/off LEDs alternately at the rate of 1 second. Use crystal frequency 16 MHz.

```
.ORG 0X00
HERE: SBI DDRD, 6
SBI PORTD, 6
CALL DELAY
CBI PORTD, 6
CALL DELAY
SBI DDRD, 7
SBI PORTD, 7
CALL DELAY
CBI PORTD, 7
CALL DELAY
RJMP HERE
DELAY:
; Outer loop: 255 x 255 x ~16 cycles = ~1s
LDI R18, $04
LOOP3:
LDI R16, $FF
LOOP2:
LDI R17, $FF
LOOP1:
; Delay ~4 cycles per iteration (2 for dec, 2 for brne)
NOP
DEC R17
BRNE LOOP1
DEC R16
BRNE LOOP2
DEC R18
BRNE LOOP3
RET
```


AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ARDUINO
UNO
ON
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
Vin
A0
A1
A2
A3
A4
A5