Assembly Examples

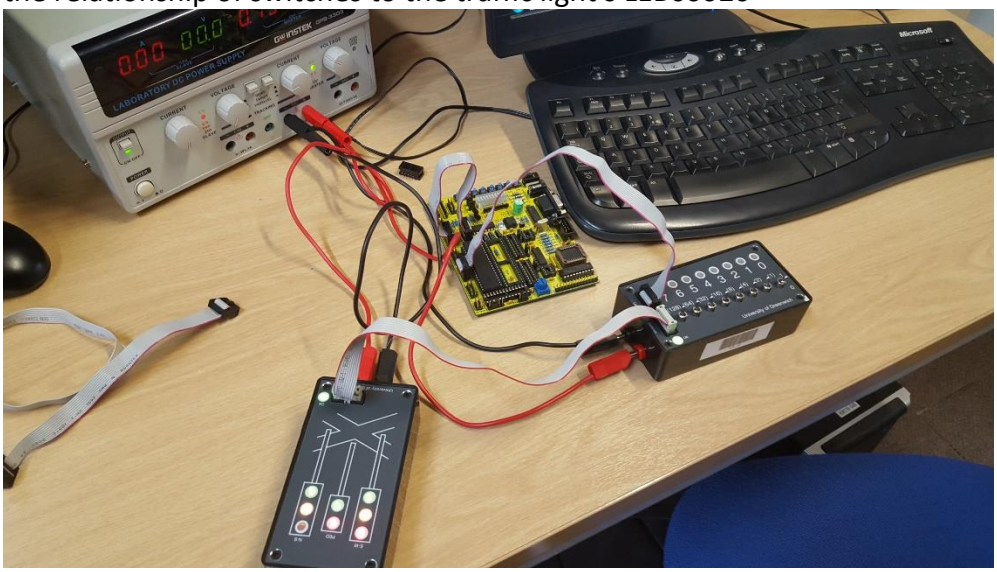
USMAN BASHARAT

00874782

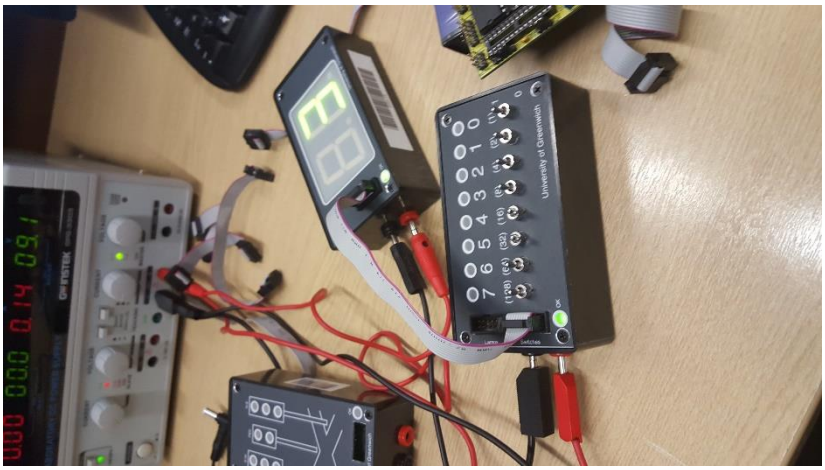
TASK 0 - Using the grey IDC10 ribbon cable (DO NOT USE THE RAINBOW RIBBON CABLE)

- Check that all the boxes have the IDC connections with the the notch facing inwards, as shown for the switch light box diagram above. (Sometime these connections pull out and are incorrectly reinserted back to front)
- How to Test IDC cables. Connect the switch light box's switches to the switch light box's LED's. There should be a one to one relationship of the switch to the LED above it. If there is not, or some LEDs do not illuminate, then normally the cable is at fault.

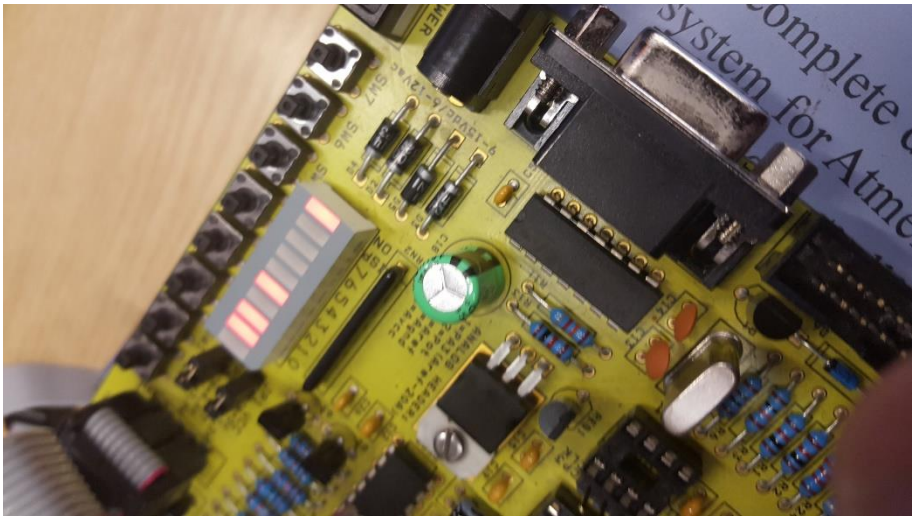
- Connect the switch light box's switches to the 'Traffic Lights' interface box and note the relationship of switches to the traffic light's LEDs0020



- Connect the switch light box's switches to the 'Dual Seven Segment Display' interface box and note the relationship of switches to the seven segment display's segments.



- Connect the STK200's on board push buttons to the switch light box's LEDs and note the relationship of the on board buttons to the switch light box's LEDs.



- Connect the switch light box's switches to the STK200's on board LEDs and note the relationship of switches to the LEDs

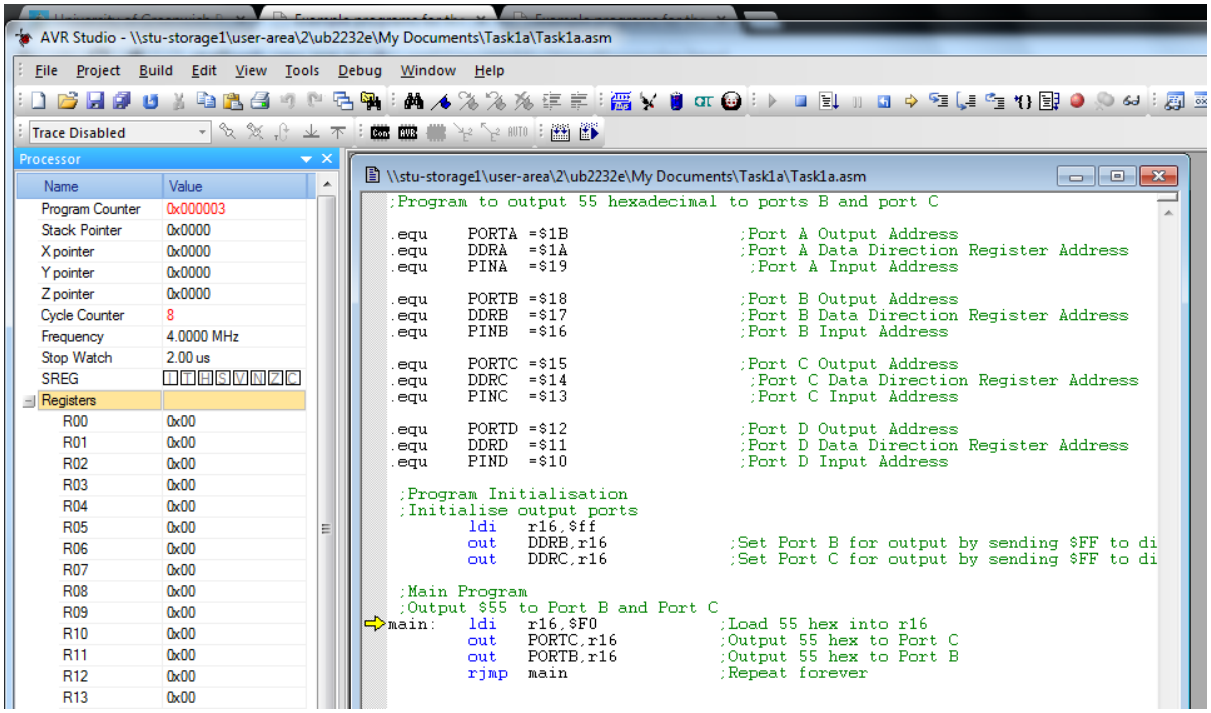
Task 1

;Program to output 55 hexadecimal to ports B and port C

```
.equ    PORTA =$1B              ;Port A Output Address
.equ    DDRA  =$1A              ;Port A Data Direction Register Address
.equ    PINA  =$19              ;Port A Input Address

.equ    PORTB =$18              ;Port B Output Address
.equ    DDRB  =$17              ;Port B Data Direction Register Address
.equ    PINB  =$16              ;Port B Input Address

.equ    PORTC =$15              ;Port C Output Address
.equ    DDRC  =$14              ;Port C Data Direction Register Address
.equ    PINC  =$13              ;Port C Input Address

.equ    PORTD =$12              ;Port D Output Address
.equ    DDRD  =$11              ;Port D Data Direction Register Address
.equ    PIND  =$10              ;Port D Input Address

 ;Program Initialisation
 ;Initialise output ports
      ldi   r16,$ff
      out   DDRB,r16          ;Set Port B for output by sending $FF to direction register
      out   DDRC,r16          ;Set Port C for output by sending $FF to direction register

 ;Main Program
 ;Output $55 to Port B and Port C
main:   ldi   r16,$F0         ;Load 55 hex into r16
      out   PORTC,r16         ;Output 55 hex to Port C
      out   PORTB,r16         ;Output 55 hex to Port B
      rjmp  main              ;Repeat forever
```

Calculation

Just add all the clocks up as shown. Adding them equals towards 8



Task 2

;Program to echo the switches attached port A to ports B and C

;Program to echo the switches attached port A to ports B and C

```
;Port Addresses
.equ    PORTA =$1B
.equ    DDRA  =$1A

.equ   PORTB  =$18
.equ   PINB   =$16

.equ    PORTC =$15
.equ   PINC    =$13

.equ   PORTD  =$12
.equ    DDRD  =$11


;Register Definitions
.def    temp   =r16             ;Temporary storage register
```

```
;Program Initialisation
;Initialise Input Ports
      ldi   temp,$00
      out   PORTC,temp        ;Set Port A for input by sending $00 to direction register
            out   PORTB,temp
; Initialise output ports
      ldi   temp,$FF        ;Set Port C for output by sending $FF to direction register
                  out   DDRA,temp
                  out   DDRD,temp
; Main Program
loop:   in    r17,PINC
      in    r18,PINB
      out   PORTD,r17
      out   PORTA, r18

      rjmp  loop  Task 3
chase2.asm
;Program to sequence LEDs on port C with delay loop

 ;Port Addresses
.equ    PORTC  =$15          ;Port C Output Address
.equ    DDRC   =$14          ;Port C Data Direction Register

;Register Definitions
.def    leds   =r0          ;Register to store data for LEDs
.def    temp   =r16          ;Temporary storage register

 ;Program Initialisation
 ;Initialise output ports
      ldi   temp,$ff
      out   DDRC,temp       ;Set Port C for output by sending $FF to direction register

 ;Initialise Main Program
      sec               ;Set carry flag to 1
      clc   leds          ;Clear LEDs

 ;Main Program
forever: out   PORTC,leds      ;Display LEDs to port C

 ;delay section of code (25.344 ms @ 1MHz) - utilises r25,r24
      ldi   r24,$C4       ;Initialise 2nd loop counter
loop2:  ldi   r25,$FF       ;Initialise 1st loop counter
loop1:  dec       r25          ;Decrement the 1st loop counter
      brne  loop1         ;and continue to decrement until 1st loop counter = 0
      dec   r24          ;Decrement the 2nd loop counter
      brne  loop2         ;If the 2nd loop counter is not equal to zero repeat the 1st loop, else
continue
```

```
;end of delay section of code

        rol    leds            ;Rotate LEDs left by 1 bit through carry flag
        rjmp   forever
```

Calculation Delay
1+[(255 * 3) – 1] = 765 one loop
1+765*255 + (255-1)*3+2 = 195840 maximum delay possible using two nested loops
195840/100000 = 195.840ms delay

Task 4
Using three loops
1+(255-1)*3+2=765
1+195840*255+(255-1)*3+2 = 49939965
4993965/1000000 = 49.939965s so 49.94s
Delay for 1.175s
1+(255-1)*3+2=765
1+765*255+(255-1)*3+2 = 195840
1+195840*6+(6-1)*3+2 = 1175058s
1175058s/1000000 = 1.175058s so 1.175s

TASK 5- Attach the switch light box's LEDs to Port C.

- Calculate the maximum delay possible with this technique @ 1 MHz clock frequency. (Answer 262.141 ms)
- Alter 'chase 4' so the delay is 150 ms.

Demonstrate the program to the tutor and upload the program, flow chart and delay calculations to your logbook.

(3*255)+(3*255*255)+(3*255*255*255)+(3*255*255*255*255) = 12734691840 micro seconds
150000/4=37500= $927C

;Program to sequence LEDs on port C using a delay subroutine

```
;Status Register Address
.equ    SREG   =$3F             ;Status Register Address

;Stack and Stack Pointer Addresses
.equ    SPH    =$3E             ;High Byte Stack Pointer Address
.equ    SPL    =$3D             ;Low Byte Stack Pointer Address
.equ    RAMEND =$25F            ;Stack Address

;Port Addresses
.equ    PORTC  =$15             ;Port C Output Address
```

```
.equ    DDRC  =$14          ;Port C Data Direction Register Address

 ;Register Definitions
.def    leds  =r0           ;Register to store data for LEDs
.def    temp  =r16          ;Temporary storage register
.def    save  =r19          ;Temporary storage register for status register
.def    YL    =r28          ;Define low byte of Y
.def    YH    =r29          ;Define high byte of Y

 ;Program Initialisation
 ;Set stack pointer to end of memory
       ldi   temp,high(RAMEND)
       out   SPH,temp        ;Load high byte of end of memory address
       ldi   temp,low(RAMEND)
       out   SPL,temp        ;Load low byte of end of memory address

 ;Initialise output ports
       ldi   temp,$ff
       out   DDRC,temp       ;Set Port C for output by sending $FF to direction register

 ;Initialise Main Program
       sec                 ;Set carry flag to 1
       clr   leds          ;Clear LEDs

;Main Program
forever: out   PORTC,leds     ;Display leds to port C
       rcall  delay          ;Call delay subroutine
       rol   leds          ;Rotate leds left by 1 bit through carry flag
       rjmp  forever         ;Continue forever

;Delay Subroutine (25.351 ms)
delay:  in    save,SREG       ;Preserve status register
       ldi   YH,high($927C)  ;Load high byte of Y
       ldi   YL,low($927C)   ;Load low byte of Y
loop:   sbiw  Y,1           ;Decrement Y
       brne  loop          ;and continue to decrement until Y=0
       out   SREG,save       ;Restore Status register
       ret                 ;Return
```

TASK 6- Attach the switch light box's LEDs to Port C and the switch light box's switches to Port A

- Alter 'chase 5' so instead of the poll being on the MSB on port A, make the poll the Least Significant Bit (LSB) on port A - i.e. switch 0 on the switch light box . Note - operating switches 7 to 1 on the switch light box should not affect the program. Hint - reflect on subnet masks from the networking course.

```
;Program to sequence LEDs on port C, uses the MSB on switches to change direction

;Stack and Stack Pointer Addresses
.equ    SPH    =$3E          ;High Byte Stack Pointer Address
.equ    SPL    =$3D          ;Low Byte Stack Pointer Address
.equ    RAMEND =$25F         ;Stack Address

;Port Addresses
.equ    DDRA   =$1A          ;Port A Data Direction Register Address
.equ    PINA   =$19          ;Port A Input Address
.equ    PORTC  =$15          ;Port C Output Address
.equ    DDRC   =$14          ;Port C Data Direction Register Address

;Register Definitions
.def    leds   =r0           ;Register to store data for LEDs
.def    temp   =r16          ;Temporary storage register
.def    chdir  =r20          ;Register determining sequence direction of LEDs

;Program Initialisation
;Set stack pointer to end of memory
      ldi    temp,high(RAMEND)
       out    SPH,temp       ;Load high byte of end of memory address
       ldi    temp,low(RAMEND)
       out    SPL,temp       ;Load low byte of end of memory address

 ;Initialise Input Ports
       ldi    temp,$00
       out    DDRA,temp      ;Set Port A for input by sending $00 to direction register

 ;Initialise Output Ports
       ldi    temp,$ff
       out    DDRC,temp      ;Set Port C for output by sending $FF to direction register

 ;Initialise Main Program
       sec                   ;Set carry flag to 1
       clr    leds           ;Clear leds

;Main Program
```

```
forever: out     PORTC,leds       ;Display leds to port C
         rcall  delay             ;Call delay subroutine
         rcall  pollswt           ;Poll switches to check direction change
         bst    chdir, 0          ;Test if negative
         brts   right             ;If switch 7= 1 chase right else if if switch 7 = 0 chase left

;Rotate leds left
left:    rol    leds              ;Rotate leds left by 1 bit through carry flag
         rjmp   forever

;Rotate leds right
right:   ror    leds              ;Rotate leds right by 1 bit through carry flag
         rjmp   forever

;Polling Subroutine
pollswt: in     chdir,PINA        ;Read switches on switch light box
         ret

;delay section of code (25.348 ms @ 1MHz) - utilises r25,r24
delay:   ldi    r24,$21           ;Initialise 2nd loop counter
loop2:   ldi    r25,$FF           ;Initialise 1st loop counter
loop1:   dec    r25               ;Decrement the 1st loop counter
         brne   loop1             ;and continue to decrement until 1st loop counter = 0
         dec    r24               ;Decrement the 2nd loop counter
         brne   loop2             ;If the 2nd loop counter is not equal to zero repeat the 1st loop, else
continue
         ret                      ;Return
```

Task 7

```
;Program to sequence LEDs on port C, uses interrupt on button SW2

;Stack and Stack Pointer Addresses
.equ    SPH    =$3E          ;High Byte Stack Pointer Address
.equ    SPL    =$3D          ;Low Byte Stack Pointer Address
.equ    RAMEND =$25F         ;Stack Address

;Interrupt control Addresses
.equ    GIMSK  =$3B          ;General Interrupt Mask Address
.equ    MCUCR  =$35          ;Machine Control Unit Control Register Address

;Port Addresses
.equ    PORTC  =$15          ;Port C Output Address
.equ    DDRC   =$14          ;Port C Data Direction Register Address

;Interrupt Vector Addresses
.equ    INT1addr=$002        ;External Interrupt0 Vector Address

;Register Definitions
.def    leds   =r0           ;Register to store data for LEDs
.def    temp   =r16          ;Temporary storage register
.def    chdir  =r20          ;Register determining sequence direction of LEDs

.org    $0000
        rjmp   reset         ;Reset vector

;Set interrupt vectors
.org    INT1addr
        rjmp   INT1          ;External Interrupt0 Vector

.org    $0015                ;Program address
;Program Initialisation
;Set stack pointer to end of memory
reset:  ldi    temp,high(RAMEND)
        out    SPH,temp      ;Load high byte of end of memory address
        ldi    temp,low(RAMEND)
        out    SPL,temp      ;Load low byte of end of memory address

;Initialise output ports
        ldi    temp,$ff
        out    DDRC,temp     ;Set Port C for output by sending $FF to direction register

;Initialise Interrupt
```

```
        ldi    temp,$80
        out    GIMSK,temp       ;Enable interrupt 0 (INT0)
        ldi    temp,$02
        out    MCUCR,temp       ;Set Interrupt to occur on a falling edge

;Initialise Main Program
        sec                     ;Set carry flag to 1
        clr    leds             ;Clear leds
        ldi    chdir,$0F        ;Set sequence direction to left
        sei                     ;Enable interrupts

;Main Program
forever: out    PORTC,leds      ;Display leds to port C 1
        rcall  delay            ;Call delay subroutine
        tst    chdir            ;Test if negative
        brmi   right            ;If chdir positive sequence right
        rol    leds             ;Rotate leds left by 1 bit through carry flag
        rjmp   forever
right:  ror    leds             ;Rotate leds right by 1 bit through carry flag
        rjmp   forever

;External Interrupt0 Service Routine
INT1:   cli                     ;Prevent any more interrupts while ISR is running
        swap   chdir            ;Flip nibbles
        reti                    ;Return and enable interrupts again

;delay section of code (25.348 ms @ 1MHz) - utilises r25,r24
delay:  ldi    r24,$21          ;Initialise 2nd loop counter
loop2:  ldi    r25,$FF          ;Initialise 1st loop counter
loop1:  dec    r25              ;Decrement the 1st loop counter
        brne   loop1            ;and continue to decrement until 1st loop counter = 0
        dec    r24              ;Decrement the 2nd loop counter
        brne   loop2            ;If the 2nd loop counter is not equal to zero repeat the 1st loop, else
continue
       ret                      ;Return
```

TASK 8

```
;Program to sequence LEDs on port C, using a look up table

;Stack and Stack Pointer Addresses
.equ    SPH    =$3E          ;High Byte Stack Pointer Address
.equ    SPL    =$3D          ;Low Byte Stack Pointer Address
.equ    RAMEND =$25F         ;Stack Address

;Port Addresses
.equ    PORTC  =$15          ;Port C Output Address
.equ    DDRC   =$14          ;Port C Data Direction Register Address

;Register Definitions
.def    leds   =r0           ;Register to store data pointed to by Z
.def    temp   =r16          ;Temporary storage register
.def    count  =r17
.def    YL     =r28          ;Define low byte of Y
.def    YH     =r29          ;Define high byte of Y
.def    ZL     =r30          ;Define low byte of Z
.def    ZH     =r31          ;Define high byte of Z

;Program Initialisation
;Set stack pointer to end of memory
        ldi    temp,high(RAMEND)
        out    SPH,temp      ;Load high byte of end of memory address
        ldi    temp,low(RAMEND)
        out    SPL,temp      ;Load low byte of end of memory address

;Initialise output ports
        ldi    temp,$ff
        out    DDRC,temp     ;Set Port C for output by sending $FF to direction register

;Main Program
reset:  ldi    ZL,low(table*2)  ;Set Z pointer to start of table
        ldi    ZH,high(table*2)
        clr    count         ;Set table position counter to zero
next:   rcall  delay         ;Call delay subroutine
        lpm                  ;Load R0 with data pointed to by Z
        out    PORTC,leds    ;and display data on port C
        adiw   ZL,1          ;Increment Z to point to next location in table
        inc    count         ;Increment table position counter
        cpi    count,14      ;and test if end of table has been reached
        brne   next          ;if not the end of the table, get next data value in table
        rjmp   reset         ;else reset Z pointer to start of table

;Delay Subroutine (25.349 ms @ 1MHz)
```

```
delay:   ldi    YH,high($FFFF)    ;Load high byte of Y
         ldi    YL,low($FFFF)     ;Load low byte of Y
loop:    sbiw   Y,1               ;Decrement Y
         brne   loop              ;and continue to decrement until Y=0
         ret                      ;Return

table:   .DB $3F,$06,$5B,$4F,$66,$6D,$7D,$6F,$77,$7C,$39,$5E,$79,$71
```