PRÀCTICA 4

Leds on, leds off. Ports de sortida i d'entrada.

En aquesta pràctica s'ensenya l'estructura i els elements lèxics i sintàctics principals que composen un programa en assemblador. Sobretot convé fixar-se en:

- les parts d'un programa assemblador típic

- a accedir al port de sortida (on s'hi posen els leds)

- a treballar amb subrutines i entendre el stack

- convé repassar totes les instruccions que s'utilitzen i entendre les seves possibilitats

- segons els registres amb els que es treballa (d'acord amb l'arquitectura mostrada), es tenen diferent diferents funcionalitats. Per evitar complexitats, aquí es treballarà amb els registres 16 en amunt (fins a 25)

Resum del material que cal per a realitzar aquesta pràctica.

- *Plataforma AVRstudio i programador PonyProg*
- *Microcontrolador ATMEGA8L i programador*
- *Placa de prototipatge*
- *8 leds*
- *RSIPs-9 (mòdul de 8 resistències, d'1KΩ pels leds)*

NOTA:

- *Recordeu que és indispensable que...*
- *Llegiu els enunciats previ a l'inici de la sessió de pràctiques*
- *ENVIEU PREVI A L'INICI DE LA PRÀCTICA L'INFORME PRELIMINAR AL PROFESSOR DE PRÀCTIQUES, que constarà dels esquemes i respostes a les preguntes teòriques*
- *Porteu impresos els* d'atacats *corresponents a la sessió*

Nota a les pràctiques.

Totes les pràctiques es realitzaran amb el microcontrolador ATmega8 o ATmega128. L'ATmega8 té l'avantatge de ser molt simple i permet introduir l'ús del microcontrolador de forma àgil. L'ATmega128 és, ja, un microcontrolador potent que permet fer aplicacions de força nivell. De fet és el nucli del RUAB1. Tot i aquesta diferència de prestacions, ambdós comparteixen arquitectura i conjunt d'instruccions, motiu pel que en el que respecte a les pràctiques és indiferent el microcontrolador emprat.

Tot i això cal tenir en compte dos aspectes fonamentals:

- *A l'inici del programa s'ha d'indicar amb una directiva el microcontrolador que s'empra.*
- *Els tipus de recursos que tenen ambdós microcontroladors són els mateixos: ports d'entrada/sortida, comptadors, protocols de comunicació, comparadors i ADCs, memòria i recursos de programació. Aquests recursos canvien en nombre i en algún aspecte de la seva funcionalitat segons el microcontrolador. Però la forma de programar-se i actuar és la mateixa. Per tant, cal fer servir sempre el manual d'ús del micrcoontorlador (que explica molt bé el funcionament del dispositiu)com a referència.*

4.1. El meu primer programa.

Introdueix el següent codi i verifica (per simulació amb AVRstudio) el funcionament del programa.

Llegeix atentament i comprèn el següent codi assemblador sobre el ATmega8.

Observacions:

- Cal entendre el codi, l'estructura i la sintaxi emprada, ja que en cada programa que es faci s'haurà de conservar.
- Observa el comportament de registres i port B de sortida. Observa, sobretot, la oscil·lació (fent la simulació) que es produeix en aquest port. Aquesta oscil·lació dels pins, quan estan connectats a leds, fa que aquests es vagin encenent i apagant.

```
; Programa 4.1: Encesa de leds (I)
        ; Joan Oliver. Octubre 2008
; Directiva de l'assemblador: definir el xip de treball
.INCLUDE "m8def.inc"

; Definició de registres.
; Aquí es defineix un registre de treball
; Es fa amb .DEF i permet anomenar els registres amb noms més fàcils de recordar
.DEF temp = R16

; Inici del codi
; El primer és definir l'adreça de començament
; Sempre que el programa recomença ho fa per aquesta adreça 0000
; Es pot començar en el power-on, per reset hardware o per timer watchdog que arriba a zero
; Aquesta adreça conté un salt a la part principal del programa.
; No cal calcular a quina adreça es salta. L'assemblador ho fa automàticament
.ORG 0x0
RJMP inici

; Ara comença el programa. S'ha de fer amb l'etiqueta on es salta. Convé començar les
; etiquetes en la primera columna de la fila,  mentre que les instruccions les posarem tabulades.
; La tabulació ajuda a llegir el codi
inici:
; Primer de tot es defineix el port D (on hi ha els leds) com a sortida.
; S'escriuen 8 1's en el registre de direcció del portD (indicant que actuen com a sortida). Així
;       - primer es carrega en un registre temporal el valor de cada pin
;               1 indica sortida, 0 indica entrada
;       - després s'envia aquest valor al port, que s'anomena DDRD
        LDI     temp, 0b11111111 ; temp <- 0xff
        OUT     DDRD, temp       ; DDRD <- temp
; a tenir en compta LDI només és vàlid pels registres r16 a r31.
; OUT escriu el contingut del registre en un port (veure cada dispositiu – aquí atmega8)
; Com que hem connectat els leds (en directa) cap a terra a través d'una resistència, aquests
; s'encendran quan posem un 1 al port de sortida
        LDI     temp, 0xff              ; temp <- 0x00
        OUT     PORTD, temp             ; DDRD <- temp

; Si ara els volem apagar, haurem d'escriure 0's al port D...
        LDI     temp, 0x00              ; temp <- 0x11
        OUT     PORTD, temp             ; DDRD <- temp

; Si ara volguéssim encendre i apagar contínuament els leds, només hauríem de posar un salt
```

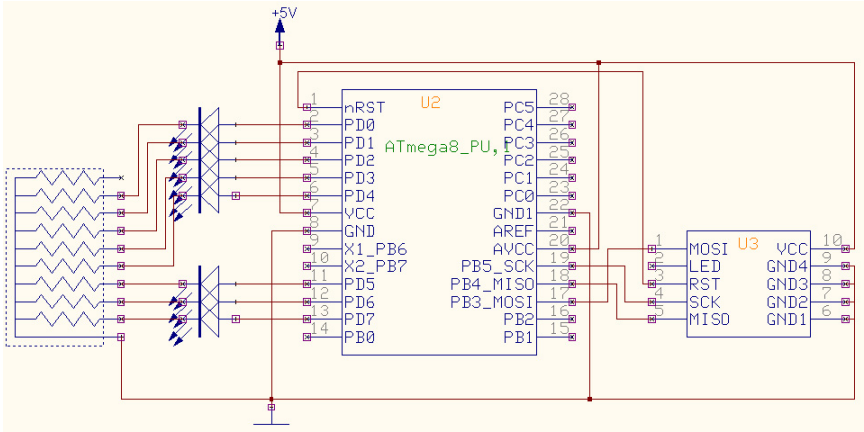
```
; a la primera instrucció LDI. Això ho fem amb un salt...
        RJMP bucle
```

Per a què funcioni S'HA DE POSAR UNA ETIQUETA a la instrucció on saltem, a la segona instrucció

bucle: LDI temp, 0xff ← MODIFICAR AQUESTA INSTRUCCIÓ

Assembleu el codi i baixeu-lo a placa, què s'observa?

Emprant la placa de prototipatge i resta de components connecteu un led a cada pin del port D del microcontrolador (realitzeu el muntatge de la figura adjunta).



TREBALL PREVI:

- Donar un diagrama de flux del programa.
- Realitzar la simulació per a comprovar-ne el funcionament

TREBALL PRÀCTIC:

Què s'observa? Per què passa això? Segur que si enlloc de mirar al led hi poseu l'oscil·loscopi al pin en tindreu la resposta... Justifiqueu la resposta amb càlculs precisos de la freqüència de funcionament.

4.2. Instruccions de salt.

Si volem veure els leds pampalluguejar, s'ha d'introduir un retard en l'encesa i en l'apagada dels leds. Això es pot fer un doble bucle anidat.

... anem a pams:

Bucles

El següent tros de programa fa un bucle incrementant, pas a pas, el contingut d'un registre:

```
.DEF cnt, r17        ; Es defineix un registre que emprarem per comptar
...
LDI cnt, 0xff              ; s'inicialitza el registre comptador
loopcnt: DEC cnt           ; s'incrementa el comptador
BRNE loopcnt               ; torna a loopcnt si no és zero, si no continua en seqüència
```

Comentaris.

BRNE és una instrucció de salt. La condició de salt depèn del càlcul de la instrucció anterior. DEC activa el bit de zero quan s'arriba a 0. Si no ho és, BRNE torna a la instrucció loopcnt. Si ho és, continua en seqüència.

El bucle triga 255*(1cicle(INC)+2cicles (BRNE))-1cicle(no salt) = 764 cicles en executar.

I si posem dos bucles anidats?

...això ens proporcionaria uns 200000 cicles de retard. Anem-ho a provar...

Introduïu aquest codi en l'assemblador

- Fixeu-vos molt bé en l'anidament de bucles que introdueix un retard important en el canvi lògic de les sortides.
- Intenteu simular-lo.
- Donat que els bucles són molt grans, difícilment aconseguireu veure un resultat 'agradable'.

Ajut: canvieu els valors 0xff de la inicialització dels comptadors per valors baixos, de forma que en pocs cicles es pugui veure com commuten les sortides

- Sobretot cal observar en la simulació, el canvi que es produeix en les sortides del port D i comprovar el funcionament dels registres comptadors. Entendre bé què passa amb els ports i els registres en el simulador és fonamental per anar comprenent el funcionament del microcontrolador i del simulador.

```
; Programa 4.2: Encesa de leds (II)
; Joan Oliver. Octubre 2008
.INCLUDE "m8def.inc"
.DEF temp = r16              ; Definició registre temporal
.DEF cnt1 = r17              ; Definició registre comptador 1
.DEF cnt2 = r18              ; Definició registre comptador 2

        .ORG 0x0
        RJMP inici
inici:  LDI   temp, 0b11111111
        OUT   DDRD, temp     ; Port D com a sortida
bucle:  LDI   temp, 0xff
        OUT   PORTD, temp    ; Leds encesos
; Retard per leds on
        LDI cnt1, 0xff       ; s'inicialitza el comptador extern
loopcnt11: LDI cnt2, 0xff    ; s'inicialitza el comptador intern
loopcnt12: DEC cnt2          ; decrement comptador intern
        BRNE loopcnt12       ; bucle intern
        DEC cnt1             ; decrement comptador extern
        BRNE loopcnt11       ; bucle extern

        LDI   temp, 0x00
        OUT   PORTD, temp    ; Leds apagats
; Retard per leds off
        LDI cnt1, 0xff       ; s'inicialitza el comptador extern
loopcnt21: LDI cnt2, 0xff    ; s'inicialitza el comptador intern
loopcnt22: DEC cnt2          ; decrement comptador intern
        BRNE loopcnt22       ; bucle intern
        DEC cnt1             ; decrement comptador extern
        BRNE loopcnt21       ; bucle extern
        RJMP bucle
```

TREBALL PREVI:

- Calculeu la freqüència exacta d'encesa dels leds.
- Realitzeu un diagrama de flux de la versió 2 del programa
- Realitzar la simulació per a comprovar-ne el funcionament

TREBALL PRÀCTIC:

- Comproveu, baixant a placa el programa, que els leds s'encenen a la freqüència calculada.

4.3. Introduint subrutines.

En el programa anterior, el bucle retard es crida dos cops, essent el programa el mateix. De poder escriure el codi només un cop, s'alliberaria espai de memòria.

El tros de programa que ho fa possible és la *subrutina*.

Una subrutina consta de les instruccions CALL (crida) i RET (retorn)

- *CALL rutina* fa un salt de programa a l'etiqueta rutina

- *RET* retorna a la posició següent des d'on s'ha cridat

Funcions de la instrucció CALL:

- Guarda l'adreça de la instrucció següent en una memòria anomenada *stack*
- Salta a l'adreça especificada per l'etiqueta, començament de l'etiqueta

Funcions de RET:

- Recupera l'adreça guardada en el *stack* i retorna en seqüència

El stack

- És una pila: la darrera dada en entrar és la primera en sortir (*Last-In, First-Out – LIFO*).
- Com que no se sap l'espai que el stack pot necessitar, es usual posar-lo en memòria començant per la darrera adreça. Així, cada cop que una nova adreça s'ha de guardar, la posició del stack decrementa. Quan es treu l'adreça, s'incrementa.
- L'ATMEGA posiciona el stack en la darrera posició de la memòria (RAMEND), i ve definit en el fitxer "*xxxdef.inc*"
- Com que els registres de l'ATMEGA treballen amb bytes, i l'adreçament de memòria es fa amb dos bytes, cal realitzar dos accessos a registres per definir el stack, ja que cal guardar el byte superior i l'inferior
- El següent exemple mostra el programa anterior, amb subrutina, i amb el stack.

Agafeu el codi següent, que implementa el mateix programa amb subrutines i...

- Reviseu el codi corresponent a la definició de l'espai de memòria on comença el stack i la subrutina *retard* (està posat en format negreta)

- Compileu-lo i simuleu-lo. Donat que els bucles provoquen un retard molt gran com per poder observar bé la simulació, canvieu momentàniament els valors d'inici dels comptadors per poder simular-lo bé.

- Agafant el programador programeu el microcontrolador i observeu el resultat en els leds.

```
; Programa 4.3: Introducció de la subrutina
; Joan Oliver. Octubre 2008
.INCLUDE "m8def.inc"
.DEF temp = r16                 ; Definició registre temporal
.DEF cnt1 = r17                 ; Definició registre comptador 1
.DEF cnt2 = r18                 ; Definició registre comptador 2

        .ORG 0x0
        RJMP inici
inici:  LDI r16, low(RAMEND)    ; Byte baix de la posició de memòria final
        OUT SPL, r16            ; SPL guarda la posició baixa de l'apuntador al stack
        LDI r16, high(RAMEND)   ; Byte alt de la posició de memòria
        OUT SPH, r16            ; SPH guarda la posició alta de l'apuntador al stack
        LDI   temp, 0b11111111
        OUT   DDRD, temp        ; Port D com a sortida
bucle:  LDI   temp, 0xff
        OUT   PORTD, temp       ; Leds encesos
```

```
        RCALL    retard          ; Crida bucle de retard
        LDI   temp, 0x00
        OUT   PORTD, temp        ; Leds apagats
        RCALL    retard          ; Crida bucle de retard
        RJMP bucle

retard: LDI cnt1, 0xff           ; s'inicialitza el comptador extern
    loopcnt1: LDI cnt2, 0xff     ; s'inicialitza el comptador intern
            loopcnt2: DEC cnt2 ; decrement del comptador intern
        BRNE loopcnt2            ; bucle intern
        DEC cnt1                 ; decrement del comptador extern
    BRNE loopcnt1                ; bucle extern
    RET                          ; es retorna de la subrutina
```

TREBALL PREVI (a entregar previ a l'inici de la primera sessió de pràctiques):

- Calculeu exactament el retard que introdueix la subrutina d'espera. A quina freqüència s'encenen els leds?
- Realitzar la simulació per a comprovar-ne el funcionament

TREBALL PRÀCTIC:

Comprovar que funciona correctament amb la freqüència d'encesa de leds esperada.