Quan crido calls es guarda la adresa a la pila, cal anar alerta que a dins de subrutina fagi els mateixos push que pops sino hi haura problemes d'adreça al ser el RET

SREG → 0x5F (0x3F)

delay: push r19
push SREG
:
pop SREG
pop r19
RET

PUSH només accepta resistres de propòsit general 0-31

Quan accedim a SREG desde in out → 0x3F *

delay Push r19
In r19, 0x3F
Push r19
:
Pop r19
out 0x3F, r19
Pop r19
RET

* canvia sino és la 0x5F

man python → a terminal

```
LDI r3, 2   ✗
```
↳ seria 1001₂ → r19

les immediates només poden accedir a les adreces 16-31

```
LDI r19, 3+2   ✓
```

el compilador utilitza ell valor 5, la CPU no → la seva instrucció de suma és Add

```
LDI r19, 2+r2   ✗
```

Faria un 2+2, no llegiria el valor de r2

```
const = 3
LDI r19, 2+const   ✓
```

És molt útil per variar programes amb el simple fet de variar una constant

```
LDI r19, 3+0b00010001   ✓
```

l'assemblador interpreta binari, hexa, octal i decimal

```
LDI r19, 'L'   ✓
```

la 'L' és un caràcter, que ocupa un byte

```
LDI r19, 'L'+2   ✓   es
```

escriuria el codi ASCII de 'N'

```
LDI r19, 'L'*2   ✓
```

multiplicar és desplaçar els bits, si que es pot, si passa el límit torna a començar?

de quan és la memòria de programa?

A LA CPU

Per sumar

ADD r3,r5 ✓ Puc utilitzar tots els registres! 0-31

ADD r3,5 ✓ suma r3 + r5

ADD r3, 45 X

CPI r16,5 } ✓
ADD r3,r16 }

Per sumar un r amb una entrada sortida, cal passarla primer a una R (~~0~~ a 32) i després sumar-lo! (in)

Sumar timer +3

✓ IN r16, TCNTO → valor del timer 0
Add r3, r16 vais sumant r3 + el valor del timer

Sumar valor posició 3000 +r3

✓ CPS r16, 3000
Add r3, r16

ADIW → suma de 16 bits → necessita 2 clocks

Per restar

SUB r3, r16 ✓ r3 → r3 − r16

SUBI r16, 0x01 ✓

SUBI r5, 0x01 ✗

↳ 16 → 31

Per accedir a 1 bit d'un registre

Posar a '1' el bit 5 de r3:

1. MOV r16, r3
   ORI r16, 0b00100000
   MOV r3, r16

2. ~~SBI P, 5~~ port entrada/sortida, bit
   No ens serveix per un registre de Propòsit General

✗ SBR r3, 5 → 0b00000101
   ↓ posa aquest bits a 1

SBR només accepta registres 16-31

SBI → P,b Posa a '1' el bit b del port P
Només del registre entrada sortida!

CBI → P,b Posa a '0'

Si vull canviar → no puc ser amb una EOR

EOR Rd, Rr

ORI Rd, K

NEG Rr

COM Rd

Imagino PORT B → 0000 0000
(5)

vull canviar el bit 5

Vull posar 0 → SBI 5,5
In 5,5 al r5 hi posa lo del port 5
Vull canviar bit → EOR 5, 0b00100000 *
OUT 5,5
~~una ORI accepta una constant!!~~

~~SIAO haurien de carregar la constant a un registre~~

* una XOR no accepta constants! primer s'ha un LDI a un registre i li carregues el 0b00100000

Quedaria

LDI r16, 0x20 → 0010 0000

IN 5, 5

EOR 5, r16

OUT 5, 5

37

IN r5, 5

Puc ser una In al resiste 5, però no un LDI

Aca vull canviar el bit 5 d'un resistre de propósit general

r3 → ORI r3, 0b00 10 0000 → No puc perquè ha de ser un r del 15 al 32

és una operació

havia de ser

imediata (com LDi)

```
LDI r16, 0b00100000
OR  r3, r16
```

si vull posar un 0 → seria una AND → 11011111

És imediate r16 - r31

SBR Rd, K crea una màscara K és el mateix que ORI

si K = 10100101 → posa a 1 els bits en els que hi ha '1'

CBR Rd, K

TIMER

pas 93

Control basic - determina mode de funcionament

OCR → dos registres de comparació → passaran coses si el valor del comptador és igual a un valor determinat

→ Mode normal → compte de 0 a 255 i quan acaba torna a 0

(?) TCNT → pot canviar el límit a on para i torna a zero (?)

→ Mode comparador → va comptant i quan arriba al límit A torna a començar

Compare output

COM0A1	COM0A0	
0	0	operació normal, OC0B → a 0
0	1	canvia el seu
1	0	quan arriba nivell de comparació es posa a 0 la sortida
1	1	es posa a 1

sortida ↑

Mode de funcionament → ~~de~~ ho diu el ~~WGM~~ WGM (bit 0 i 2)

pag 106

CTC → mode comparació OCRA → límit on arriba counter

Com surt el clock del counter? Ho diu els CS.

0 0 0 No hi ha clocs
0 0 1 → cloc 16MHz
0 1 0 → 16MHz/8
~~1 0~~
0 1 1 → 16MHz/64
⋮
1 1 0 → clock ve de pota externa → clock al ~~falling~~ edge
1 1 1 → " al rising edge

Pràctica 5

comunicació sèrie:

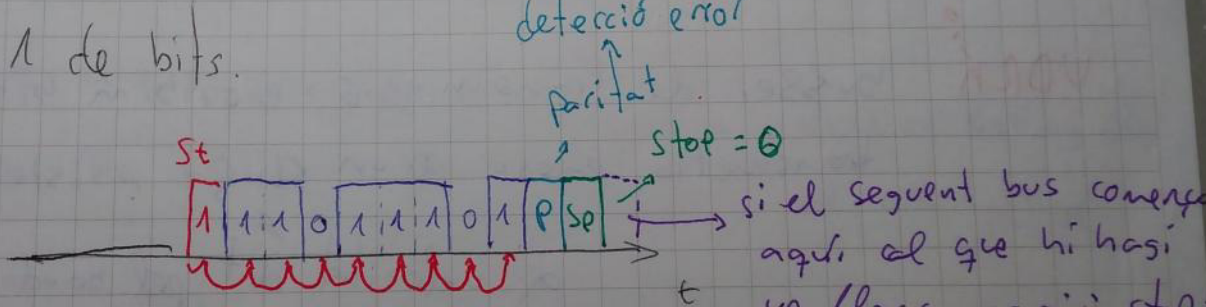
Tenim 2 màquines. Per comunicar-se és necessari enviar bits.

es pot ser → sèrie
→ paral·lel → 8 cables per exemple

És fonamental un clock

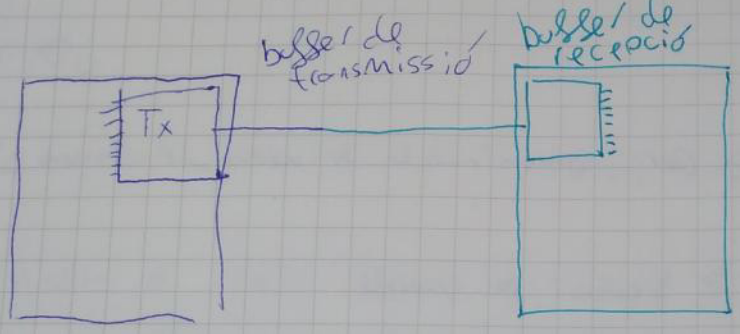
- En sèrie reduïm cables però necessitem 8 clocks
- Amb 8 cables, potser hi ha retards (pel tema llargada dels cables) entre els bits.

Ara eliminem el rellotge per tant tenim 2 cables, GND i 1 de bits.



1- bit d'start el programa sap a quina freqüència arriba cada bit és a dir, cada quan canvia el bit. (prèviament hem pactat les velocitats) Sempre que s'envia dada, es posa un 1 al principi per sincronitzar (amb 8 bits no es podria desfasar fo suficient tant com per saltar-me bits

baud rate



ES troba a entrada/sortida

registre de configuració port sèrie (vel a la que es llegeixen o escriuen dades en sèrie) UBBRn

XCKn per un rellotge extern

UDRn buffer de transmissió → escribim bit a transmetre, passa per un shift register a velocitat dit pel baud rate

cal esperar a que es transmeti tot del shift register abans de tornar a escriure a UDRn

Hi han indicadors per saber si s'ha acabat la transmissió

buffer de recepció

en aquest cas el registre es diu UDR1 també

Són dos registres diferents però és la mateixa adreça ~~de memòria~~. Comparteix adreça perquè no té ~~són dos~~ sentit llegir el registre de transmissió o escriure en el de recepció.

Accedeixes a un registre o un altre en funció de si llegeixes o escrius.

RxD1 per on entre el bus de dades en sèrie

Al ser una lectura de UDR1, aquest es buida. si abans de llegir, hi entra un altre bus.

UDR1 és una cua! pot guardar 2 bytes.

si entra el tercer bus, malament, en matxaca un de la cua. cal anar llegint UDR1 per buidar-la

sempre es troba a la Rutina Rx fins que arriba algo quan arriba surt i entra a Tx i el transmet.

PRÀCTICA 6

interrupcions → és com una subrutina, però, quan es crida?

Ho fa el hard per si mateix (els perifèrics) no el loop.

és un codi que s'executarà quan es produeixin certes situacions del hard.

Exemple Rx → ha d'anar mirant si un bit està activat, sent-ho sovint, si està activat → carrega a UART0

Registre d'estat → flags de les interrupcions global → si està a 1 → s'activen totes, llavors a cada perifèric hem d'activar quines volem

~~A part, hi ha interrupcions per cada perifèric.~~

PORT SÈRIE → Interrupcions:

→ 1 recepció: 2 de transmissió

Per què vull rutina d'interrupció de transmissió?

per avisar quan podem tornar a transmetre

- Avisa quan UART0 està buit
- Avisa quan s'ha acabat d'enviar

totes les rutines d'interrupció s'activa i salta, no es manté.

Mentre s'executa la rutina d'interrupció. paral·lelament es poden tornar a activar els flags.