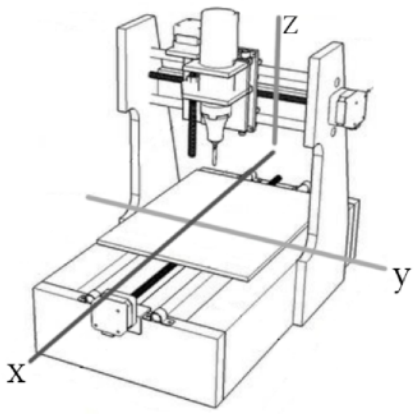
Prova di progetto - 5 Febbraio 2024

Un processore z64 comanda una macchina utensile costituita da un TRAPANO a colonna e da un PIANO su cui viene poggiata un'asse di legno da forare. Il PIANO può spostarsi lungo gli assi x e y, per allineare il pezzo di legno al TRAPANO . Il TRAPANO può muoversi lungo l'asse z per procedere alla foratura. Un SENSORE posto sul piano informa lo z64 quando la punta del trapano ha trapassato il pezzo di legno da forare.

SENSORE opera in maniera asincrona. PIANO e TRAPANO operano in modalità sincrona e sono dotati di motori passo passo che consentono di effettuare uno spostamento lungo uno degli assi di 1mm : ogni volta che i dispositivi vengono avviati, essi effettueranno uno spostamento pari a 1mm. Degli appositi registri di interfaccia consentono di indicare in quale direzione lungo l'asse corrispondente un motore deve effettuare lo spostamento passo passo. Il sistema è schematizzato nella seguente figura:



Un vettore buchi memorizzato nella memoria dello z64 individua le posizioni in cui è necessario effettuare i buchi sull'asse. Ciascun elemento del vettore è una word: gli 8 bit più significativi identificano la coordinata x, mentre gli 8 bit meno significativi individuano la coordinata y. Sia la coordinata che la coordinata sono degli interi senza segno.
All'accensione del sistema, PIANO si trova alle coordinate . Anche TRAPANO si trova alla coordinata . Poiché PIANO può essere posizionato manualmente ad una altezza variabile, **è necessario gestire a software il ritorno di TRAPANO alla posizione iniziale dopo ciascun buco**.

Il vettore buchi deve essere configurato per effettuare i buchi nelle posizioni (3mm,4mm) (6mm,8mm) e (2mm,2mm).

Realizzare:
Le interfacce dei dispositivi TRAPANO , PIANO e SENSORE ;

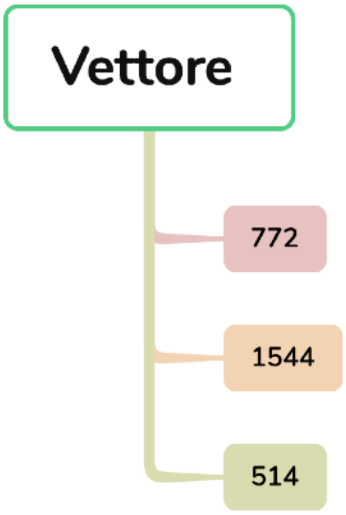
S V O L G I M E N T O

Prima di procedere con la relativa spiegazione delle interfacce utilizzate si procede nel fornire delle assunzioni di progetto.

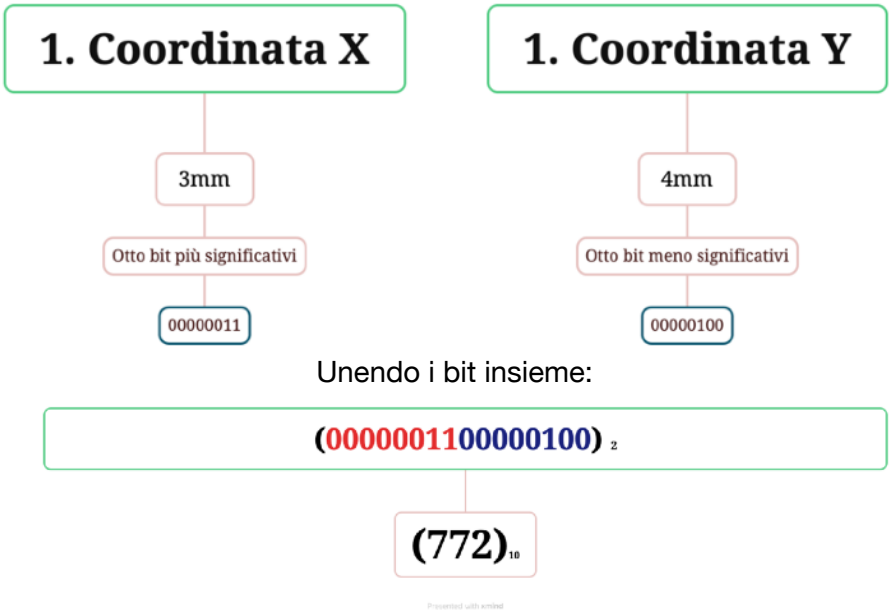
La specifica richiede che esista un piano (x,y) su cui appoggiare un asse in legno a cui verrà applicata la foratura; La foratura dovrà essere effettuata da parte del trapano, che si muove lungo l'asse z, su un punto preciso di questo piano: il che implica dire che, come già specificato dal testo, per procedere alla foratura del piano e nel contempo attivare la periferica trapano bisogna conoscere l'esatta coordinata corrispondente agli assi x ed y su cui giace il piano, solo dopo che quest'ultimo è sistemato si può procedere alla foratura ed eventualmente notificare al processore l'avvenuta della stessa.

Allo stesso tempo la specifica ci fornisce tramite un vettore in memoria delle coordinate corrispondenti al punto esatto su cui verrà applicata la foratura sul piano.

Il vettore ha una capienza di tre elementi ciascuno composto da 16 bit - stiamo parlando di un vettore di word:



Il vettore viene volutamente inizializzato con le seguenti cifre decimali: 772, 1544 e 514.
Il motivo per il quale viene effettuata questa assegnazione è strategico: stiamo parlando di un numero a 16 bit di cui i primi 8 più significativi rappresentano il valore della coordinata x mentre gli ultimi 8 meno significativi rappresentano la coordinata y.



In questo modo abbiamo ricavato il primo valore corrispondente alla prima cella del vettore. Il medesimo procedimento è del tutto analogo: per ricavare il secondo ed il terzo valore del vettore si suddivide il primo byte dal secondo byte, si analizza la corrispettiva parola binaria, si unisce in un'unica parola e si converte in decimale. Per motivi di spazio il procedimento per gli ultimi due valori non è stato illustrato.
Il vantaggio però è notevole: possiamo permetterci di effettuare un'operazione di "Bit-Extracting" su ciascun elemento del vettore.
Verrà analizzata la parola finale.
Esempio: supponiamo che esista una variabile Assembly word già dichiarata con il nome di "numero".

Questa variabile al suo interno possiede il valore 772 che ricordiamo essere un numero binario che entra perfettamente in una parola a 16 bit. Non c'è overflow:

0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0

Durante l'implementazione del codice potrà servirci l'operazione di "estrazione di bit": ossia potremmo voler essere interessati a conoscere il valore degli 8 bit più significativi di questa parola oppure gli 8 bit meno significativi.
Va da sé che se attuata questa strategia non c'è verso: possiamo solo estrarre gli 8 bit corrispondenti alla coordinata x e quelli relativi alla coordinata y.
La variabile numero è un'intera parola binaria e non ci basta: vogliamo conoscere la coordinata x!
Per conoscere la coordinata relativa all'asse x del piano bisogna prelevare gli 8 bit più significativi (il byte più significativo) di numero ed effettuare questa estrazione con un'istruzione nell'ISA dello Z64 che si chiama AND.
Questo è un tipico esempio di operazione Bit-Wise.
Ovviamente l'ISA dello Z64 prevede che dopo il nome dell'istruzione venga specificata una dimensione (specificata come suffisso) dei dati che stiamo trattando e poiché i dati trattati sono di dimensione 2 byte utilizzeremo una word, pertanto il suffisso successivo al nome dell'istruzione sarà la lettera w.
Il primo parametro della funzione è una ***maschera di bit*** impostata ad 1 per tutti quanti i bit che vogliamo conoscere della parola originale: nel nostro esempio siamo interessati a conoscere gli 8 bit più significativi, pertanto la ***maschera di bit*** sarà la seguente:

1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0

Che in esadecimale è corrispondente a 0xff00.

Pertanto possediamo tutto il necessario per effettuare questa estrazione. L'istruzione finale sarà:

andw $0xff00, numero

A questo punto per conoscere il valore della coordinata y effettuiamo l'estrazione degli 8 bit meno significativi e l'istruzione **sarà quasi analoga**:

andw $0x00ff, numero

Effettuate le dovute premesse torniamo alla specifica.

Trapano sarà un'interfaccia molto semplice in modo particolare **"interfaccia booleana"** di acceso/spento perché dovrà attivare il trapano o conseguentemente disattivarlo dopo aver applicato la foratura.

Piano sarà un'interfaccia di **output in Busy waiting**: Il motivo per il quale esiste anche il Busy waiting è perché all'interno sarà presente un registro in cui sarà specificata la *direzione* di spostamento e un registro in cui sarà specificata la *dimensione* di spostamento.
In un primo ciclo di Busy waiting la dimensione e la direzione di spostamento sarà specificata per la coordinata x mentre in un secondo ciclo sarà specificata per la coordinata y. Essendo i numeri del vettore tutti positivi e senza segno è evidente che, nel piano x y fornito dalla specifica, gli spostamenti che verranno effettuati lavoreranno solo sulla zona del primo quadrante.
Per scelta di progetto è stata applicata una codifica per la *direzione* di spostamento:

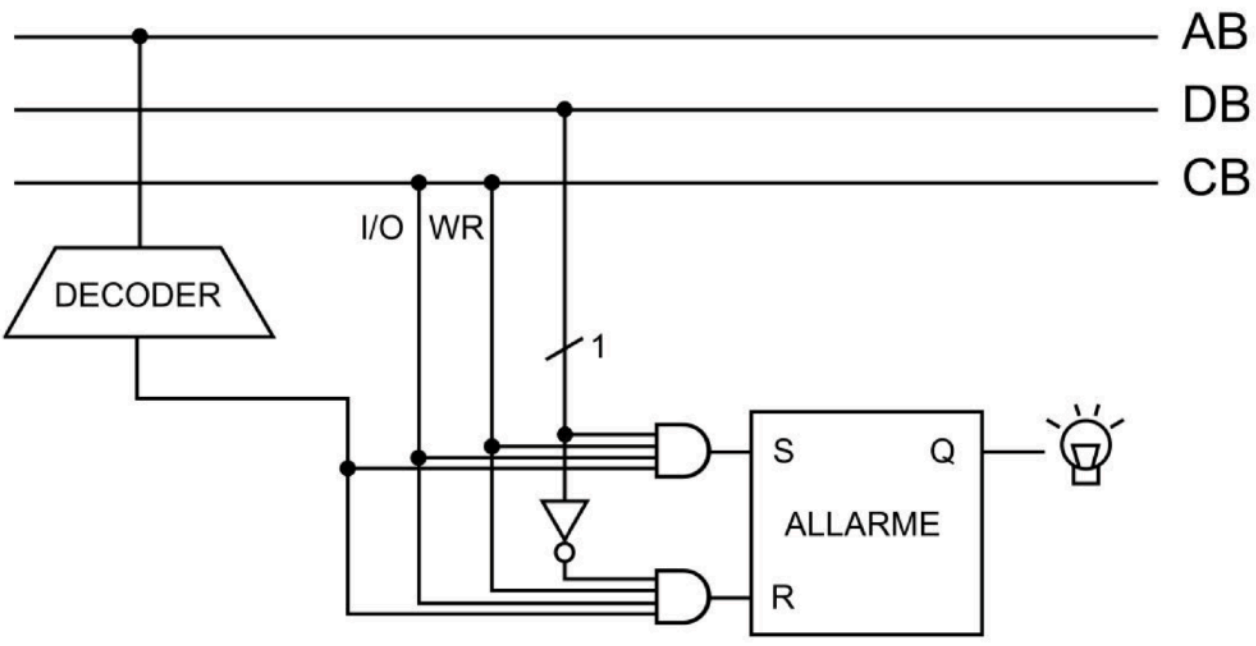
- Valore 1 per lo spostamento lungo l'asse X;
- Valore 0 per lo spostamento lungo l'asse Y;

Pertanto questa interfaccia avrà internamente due registri.

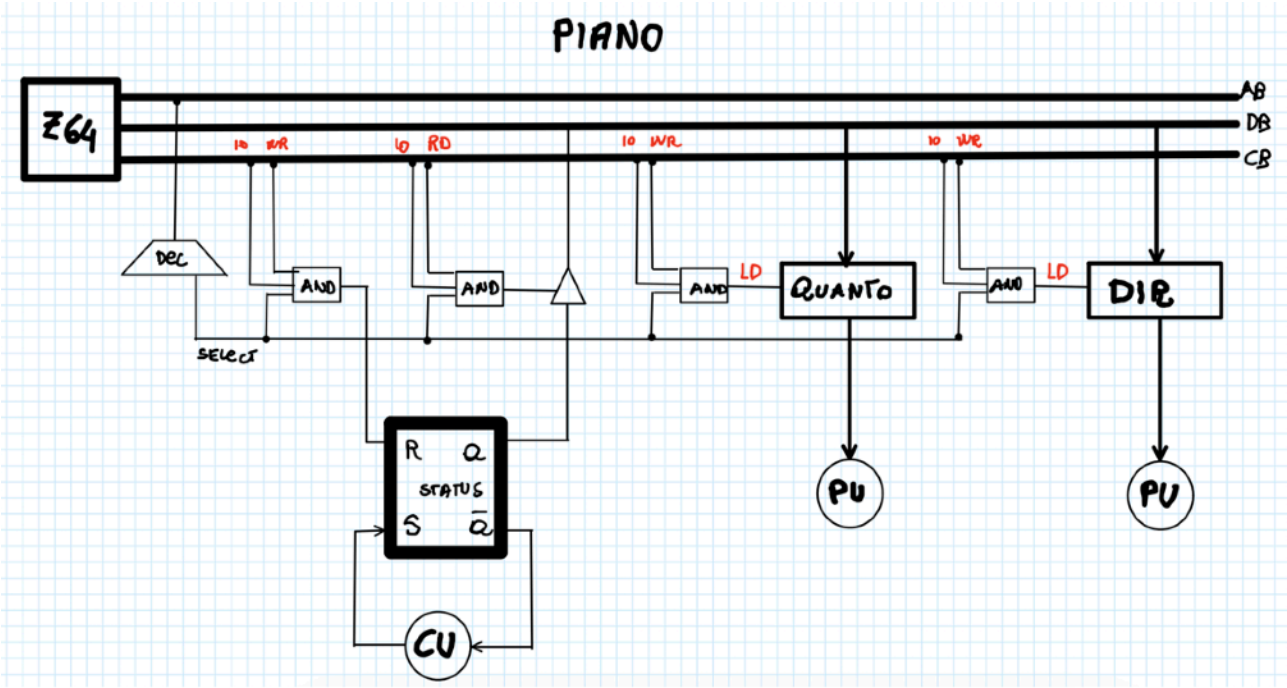
Sensore sarà un'interfaccia semplice per gestire le interruzioni vettorizzate ma non dovrà essere attivata esplicitamente perché dovrà semplicemente notificare il momento in cui la coordinata sarà trapanata.

Di seguito vengono mostrate le varie interfacce.

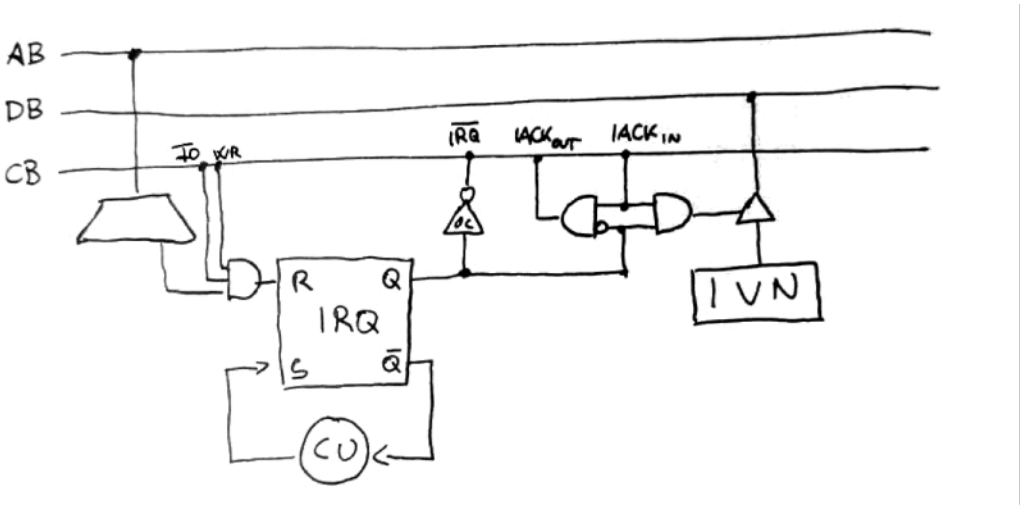
TRAPANO



PIANO



SENSORE



Di seguito viene mostrato il codice Assembly.

```
.org 0x800
.data

        .equ $STATUS_PIANO, 0x1
        .equ $dir_piano, 0x2
        .equ $quanto_piano, 0x3
        .equ $irq_sensore, 0x4
        .equ $allarme, 0x5 #è il flip flop che accende/spegne l'allarme per attivare trapano
        app: .word 0
        num_da_decifrare: .word 0
        x: .byte 0
        y: .byte 0
        buchi_trapanati: .byte 0
        buchi: .word 772, 1544, 514

.text
main:
        sti
        xorq %r15,%r15 #r15 = 0
        movq $buchi, %rdx
 .ciclo:
        movw (%rdx, %r15, 2), num_da_decifrare
        movw num_da_decifrare, app
        #app = num_da_decifrare
        andw $0xff00, app #bit-extracting, estraggo il byte più significativo
        andw $0x00ff, num_da_decifrare #bit extracting, estraggo il byte meno significativo
        movb app, x #in x coordinata X
        movb num_da_decifrare, y #in y coordinata Y

        #ora voglio spedire queste coordinate al piano. Prima x
        outb %al, $STATUS_PIANO #comunico alla periferica che voglio trasferire un dato
        movb x, %al #carico la coordinata
        outb %al, $quanto_piano #spedisco il dato sul registro "quanto"
        movb $1, %al #spostamento lungo l'asse x (1) = OPCODE
        outb %al, $dir_piano #spedisco lo spostamento sul registro "dir"
   .bw:
        inb $STATUS_PIANO, %al
        btb $0, %al
        jnc .bw
        #il dato x è sicuramente stato scritto (con il relativo spostamento sull'asse).

        #ora voglio spedire le coordinate y.
        outb %al, $STATUS_PIANO #comunico alla periferica che voglio trasferire un dato
        movb y, %al
        outb %al, $quanto_piano
        movb $0, %al #spostamento lungo l'asse y (0)
        outb %al, $dir_piano
   .bw1:
        inb $STATUS_PIANO, %al
        btb $0, %al
        jnc .bw1
        #il dato y è sicuramente stato scritto (con il relativo spostamento sull'asse).

        #a questo punto bisogna trapanare.
        call trapano
```

```
.label:
    call riposiziona
  .last:

    addq $1, %r15
    cmpq $3, %r15
    jnz .ciclo
    cmpb $3, buchi_trapanati
    jz .fine


    trapano:
        outb %al, $allarme #attivo il trapano tramite interfaccia booleana e si attiva il driver
        jmp .label

    .fine:
        hlt

    riposiziona: #riposiziono il trapano all'inizio (0,0): mi sposto indietro della stessa quantità.
        movb x, %al
        movb y, %cl
        shlb $1, %al #x * 2
        shlb $1, %cl #y * 2
        subb %al, x # x = x - ( x * 2) = 3 - ( 3 * 2) = -3 (ritorna indietro di -3)
        subb %cl, y # y = y - ( y * 2) = 4 - ( 4 * 2) = -4 (ritorna indietro di -4)
    #ora passiamoli questi dati-> in sostanza se mi sono spostato di N torno indietro di -N.

    outb %al, $STATUS_PIANO #comunico alla periferica che voglio trasferire un dato
    movb x, %al
    outb %al, $quanto_piano
    movb $1, %al #spostamento lungo l'asse x (1)
    outb %al, $dir_piano
.bw2:
    inb $STATUS_PIANO, %al
    btb $0, %al
    jnc .bw2
    #il dato x è sicuramente stato scritto (con il relativo spostamento sull'asse).

    #ora voglio spedire le coordinate y.
    outb %al, $STATUS_PIANO #comunico alla periferica che voglio trasferire un dato
    movb y, %al
    outb %al, $quanto_piano
    movb $0, %al #spostamento lungo l'asse y (0)
    outb %al, $dir_piano
.bw3:
    inb $STATUS_PIANO, %al
    btb $0, %al
    jnc .bw3
    #il dato y è sicuramente stato scritto (con il relativo spostamento sull'asse).
    jmp .last
```

```
.driver 0
        outb %al, $irq_sensore #zittisco la richiesta di interruzione
        addb $1, buchi_trapanati #notifico i buchi trapanati
        iret
```

L'assunzione che è stata effettuata è che una volta che il singolo valore della coordinata X sia stato scritto nel registro di interfaccia della periferica PIANO con il relativo OPCODE, il piano proceda a muoversi lungo l'asse X, solo dopo si potrà ricaricare un nuovo dato nel registro di interfaccia "quanto" relativo alla coordinata Y, quindi fondamentalmente si può caricare un solo valore per volta dentro il "quanto" con la relativa direzione determinata dal codice operativo prefissato (vedi sopra).

Inoltre TRAPANO è un'interfaccia di acceso/spento: si presume che una volta che sia stato attivato il trapano automaticamente parta il driver che notifichi la corretta foratura.

Il motivo di svariati cicli di Busy Waiting è dettato dal fatto che PIANO essendo un'interfaccia programmabile bisogna comunque attendere che l'unità di processamento processi i dati che il processore spedisce.