Istituto Superiore “Silvio D’Arzo”

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Cognome** | **Baroni** | **Nome** | **Mirco** | **Matricola** | **38050808** | **Classe** | **5^** |
| **Sezione** | **B** | **Indirizzo** | **ITI** | **Specializzazione** | **Elettronica** | **N° PC** | **4** |
| **Banco strum.** | **BS2** | **Laboratorio** | **Lab. Elettronica 2** | **Materia** | **Lab. TDP, Sistemi, ecc.** | **N° documento** | **1** |
| **Percorso/Nome\_file:** | | | | **Allegati:** | | | |

Tipo documento: Relazione tecnica dell’esperienza del Borrow

Progetto: Programmare il PIC in modo che effettui una sottrazione di 2 valori dati in due registri

Obbiettivo: studiare il comportamento dei flag C, DC e Z del registro STATUS, dopo che si è effettuata la sottrazione di due valori che abbiamo assegnato tramite i pulsanti collegati ai bit dei 2 registri usati per effettuare il calcolo.

Schema a blocchi o immagine del progetto:

Microcontrollore PIC16F877A

(circuito integrato nel quale è contenuto il programma)

**PORTD (Byte minuendo)**

**Ingressi PORTD (Byte minuendo)**

**PORTA (bit 0: flag Z, 1: DC, 2: C)**

DC

C

Z

STATUS (registro nel quale sono contenuti i flag Z (bit 2), flag DC (bit 1 ) e flag C (bit 0)

**Ingressi PORTC (Byte sottraendo)**

**PORTC (Byte sottraendo)**

**PORTB (Byte risultato sottrazione)**

**PORTE**

**PORTE** (bit 0, utilizzato per passare da una fase all’altra nell’effettuare la sottrazione) con pulsante

Uscite PIC

Ingressi PIC

Tabella dei componenti e degli strumenti utilizzati nel progetto:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Q.tà** | | **Descrizione dei componenti e degli strumenti utilizzati** | **Articolo dei comp. e strum. utiliz.** | **Caratteristiche dei comp. e strum. utilizz.** |
| **I** | **O** |
| **5** | **5** | **Diodi rettificatori** | **1N4007** | **VR=1000V, IO=1 A, VFM=1V** |
| **1** | **1** | **Connettore** | **X** | **A vite, 2 pin** |
| **30** | **30** | **Resistori** | **X** | **Pmax: ¼ W; R1=8,2 MΩ; R2=4,7 KΩ;**  **R3=270 Ω; R4:R19=4,7 KΩ;**  **R20:R30=150 Ω.** |
| **4** | **4** | **Condensatori Ceramici** | **X** | **C1,C2,C5,C7=100nF;**  **C3,C4=22pF.** |
| **2** | **2** | **Condensatori Elettrolitici** | **X** | **C9=2200µF;**  **C11=220µF** |
| **2** | **2** | **SW DIP 8** | **X** | **Blocco di 8 interruttori** |
| **1** | **1** | **Microcontrollore** | **Microchip** | **PIC16F877A, PDIP, 40 pin** |
| **1** | **1** | **Quarzo** | **X** | **f=8MHz** |
| **2** | **2** | **Pulsanti** | **X** | **Interruttore passo-passo** |
| **1** | **1** | **Regolatore di tensione** | **X** | **Vi max=35V, Vomax=5V, Imax=1 A** |
| **11** | **11** | **Diodi led** | **X** | **Rossi: Vmax=1,8V, Imax=15mA** |
| **1** | **1** | **Diodi led** | **X** | **Verde: Vmax=2V, Imax=15mA** |
| **1** | **1** | **Scheda di programmazione dei µC** | **EasyPIC3** | **X** |

Teoria e calcoli di progetto:

In questa esperienza noi dobbiamo creare un programma che permetta di fare la sottrazione tra due valori binari (che diamo noi tramite dei pulsanti), riportando il risultato di questa operazione all’esterno per verificare la correttezza del calcolo e per conoscere il valore del risultato della sottrazione. Inoltre, dopo che si è effettuata la sottrazione tra quei due valori, dobbiamo verificare lo stato logico dei flag Z, DC e C del registro STATUS, siccome il nostro scopo è quello di imparare il funzionamento di questi flag (cioè la variazione dello stato logico di essi, a seconda delle varie operazioni, in questo caso sottrazioni) a seconda del risultato che otteniamo. Lo stato logico di questi, dopo aver verificato eventuali variazioni di stato logico di essi, devono successivamente essere riportati all’esterno tramite 3 bit di un registro, così possiamo verificare le variazioni di stato logico di questi flag dopo la sottrazione, rispetto a quello dato all’inizio del programma. Questo è il riassunto della funzione che deve svolgere il programma del PIC: adesso spiegherò, in modo più dettagliato, il programma. Prima di spiegare il programma, voglio farvi capire il motivo per il quale ad ogni registro che ho utilizzato ho inserito un certo valore all’inizio del programma:

-STATUS => 0X3B: ho settato questo registro all’inizio del programma in modo da evitare problemi di malfunzionamento del PIC fin dall’inizio del programma ed, inoltre, ho settato i flag (cioè i bit 0, 1 e 2 di questo registro) in questo modo:

- il bit 0 di questo registro, cioè il flag C, lo settiamo ad “1” logico perché lavora in logica negativa, siccome non è un riporto ma è un prestito e, perciò, nel caso in cui avvenga un prestito, lui me lo setta a “0” logico, indicandomi il fatto che è avvenuto il prestito del teorico 9° bit di un registro speciale (in questo caso è un borrow del byte);

- il bit 1 di questo registro, cioè il flag DC, lo settiamo ad “1” logico per lo stesso motivo spiegato per il flag C, ma il bit che consideriamo, per questo bit, è il bit 4 di un registro speciale e, perciò, va a “0” logico questo flag quando avviene il riporto del bit 4 di un registro speciale (in questo caso è un borrow del nibble meno significativo);

- il bit 2 di questo registro, cioè il flag Z, lo settiamo a “0” logico, siccome esso si attiva ad “1” logico quando il risultato di un’operazione algebrica o di un’istruzione da come valore zero;

-ADCON1 => 0X07: ho inserito questo valore nel registro ADCON1 per far sì che io converta i bit del registro PORTA da ingressi analogici ad I/O digitali;

-INTCON => 0X00: ho inserito questo valore nel registro INTCON perché in questo programma non utilizzo nessun interrupt e perciò li disabilito tutti, per evitare malfunzionamenti;

-OPTION\_REG => 0XC0: ho settato questo registro in modo che esso non conti “l’intervento” di alcuni interrupt ed, inoltre, lo setto in modo che le resistenze di pull-up interne presenti nel registro PORTB vengano disattivate;

-TRISA => 0XF8: settiamo questo registro in modo che i bit 0, 1 e 2 del PORTA siano uscite, siccome dobbiamo inserire su di essi gli stati logici dei flag dello STATUS (flag Z, DC e C), mentre gli altri, invece, sono inutilizzati e, perciò, li settiamo come ingressi (e, questi, sull’hardware, verranno collegati direttamente a massa o a VCC, per far si che non siano pin flottanti);

-TRISB => 0x00: setto ogni bit del registro PORTB come uscita, siccome noi dobbiamo riportare all’esterno il valore del risultato della differenza;

-TRISC => 0XFF: setto tutti i bit del registro PORTC come ingressi, siccome noi dobbiamo inserire in quest’ultimo il valore del sottraendo della differenza;

-TRISD => 0XFF: anche il registro PORTD ha tutti i bit come ingressi, siccome dobbiamo inserire in esso il valore del minuendo della differenza;

-TRISE => 0X07: questo registro è settato in modo che i bit 0, 1 e 2 del PORTE siano ingressi (utilizzando solo il bit 0 per fare il passaggio da una fase all’altra nella sottrazione, mentre gli altri 2, inutilizzati, li colleghiamo sull’hardware a massa, per evitare che diventino flottanti) mentre gli altri bit siano a stato logico “0” per evitare malfunzionamenti (non sono né ingressi né uscite, sono pin che danno funzioni al PIC);

-PORTA => 0X03: ho inserito questo valore in questo registro sapendo che i bit dal numero 3 al numero 5 non si possono scrivere perché ingressi (e i bit 6 e 7, invece, non sono presenti in esso) mentre i bit 0, 1 e 2 di esso li ho settati in modo che abbiano lo stesso stato logico dei rispettivi flag che rappresentano [il bit 2 del PORTA rappresenta il bit 2 dello STATUS (il flag Z), il bit 1 rappresenta bit 1 dello STATUS (il flag DC) ed il bit 0 rappresenta il bit 0 dello STATUS (flag C)];

-PORTB => 0x00: ho azzerato il registro in modo che all’inizio io sia sicuro che il byte nel quale devo inserire il risultato della sottrazione sia a zero, appena alimentiamo il circuito;

-PORTC => 0X00: ho azzerato il registro PORTC, nonostante tutti i bit di esso siano ingressi, per evitare malfunzionamenti;

-PORTD => 0X00: ho azzerato il registro PORTD, nonostante tutti i suoi bit siano ingressi, per lo stesso motivo espresso per il PORTC;

-PORTE => 0x00: ho azzerato il registro PORTE, nonostante i bit da 0 al 2 siano ingressi, per far sì di evitare qualsiasi malfunzionamento, dato anche dai bit dal 3 al 7 non sono usati;

-M\_LA\_1, M\_LA\_2, M\_LA\_3: sono dei registri generali (rispettivamente di indirizzo 20h, 21h e 22h) che utilizzo per creare un ritardo di circa 1 s nell’eseguire il passaggio da una fase all’altra nel calcolare la sottrazione tra il valore che ho inserito (tramite i pulsanti collegati, ognuno di loro, ad un bit) nel PORTD e quello inserito nel PORTC (sempre tramite i pulsanti collegati, ognuno di loro, ad un bit);

-VAL\_PD e VAL\_PC: sono 2 registri generali, rispettivamente, di indirizzo 23h e 24h, che utilizziamo per memorizzare il valore che inseriamo, rispettivamente, in PORTD ed in PORTC, in modo che io li utilizzi per fare la differenza tra il valore di PORTD e quello di PORTC ed, inoltre, per far sì che il valore non vari dopo che sono passato da una parte di programma all’altra.

Il programma, in modo più dettagliato, funziona in questo modo:

il microcontrollore, appena il circuito verrà collegato all’alimentazione, setterà gli ingressi, le uscite ed i registri che utilizziamo nella configurazione di partenza. Dopo che il PIC ha effettuato quello che abbiamo appena descritto, egli attende, leggendo i bit di esso, l’inserimento del valore, binario, che dobbiamo inserire (tramite i pulsanti, ognuno collegato ai bit del registro per il minuendo) nel registro PORTD (cioè il registro dove inseriamo il valore del minuendo) e, successivamente, il valore viene memorizzato nel registro definito con la label VAL\_PD. Dopodiché, il PIC attenderà che noi premiamo il pulsante collegato al bit 0 del registro PORTE; fin quando non l’abbiamo premuto, il µC legge il valore che inseriamo in PORTD e lo memorizza nel registro definito dalla label VAL\_PD. Quando premerai e, successivamente, rilascerai il pulsante collegato al bit 0 del PORTE, il PIC passerà da questa fase a quella in cui egli leggerà il valore inserito nel registro PORTC e, successivamente, memorizzerà quest’ultimo nel registro definito dalla label VAL\_PC (i valori, binari, che dai al registro PORTC li ottieni tramite i pulsanti collegati, ognuno, ad un bit di questo registro e lo stesso vale anche per il PORTD)(1) (vedi appunti in fondo). Dopo che il µC ha effettuato la lettura del valore che è stato inserito in PORTC e memorizzato quest’ultimo in VAL\_PC, egli attenderà che tu prema e rilasci il pulsante collegato al bit 0 del PORTE continuando a leggere il valore che viene inserito in PORTC e memorizzare quest’ultimo in VAL\_PC. Dopo che tu hai premuto e rilasciato il pulsante collegato al bit 0 del registro PORTE, egli passerà da questa fase a quella in cui il µC dovrà eseguire la sottrazione tra il valore del PORTD (il minuendo), utilizzando il registro definito dalla label VAL\_PD, e quello del PORTC (il sottraendo), utilizzando il registro definito dalla label VAL\_PC. Il microcontrollore effettua la sottrazione(2) (vedi appunti) e, successivamente, riporterà al registro PORTB il valore (in binario) del risultato della sottrazione che ha appena effettuato e, successivamente, legge lo stato logico dei flag Z, DC e C del registro STATUS, per vedere se sono variati rispetto a quello che avevano prima della sottrazione. Dopodiché, noi dobbiamo settare lo stato logico dei bit 0, 1 e 2 del registro PORTA allo stesso stato logico del flag che rappresentano (bit 0 PORTA rappresenta flag C dello STATUS, bit 1 il flag DC ed il 2 il flag Z dello STATUS), perché così possiamo vedere come sono variati gli stati logici dei flag del registro STATUS dopo che il PIC ha effettuato la sottrazione tra i due valori. Dopo che il PIC16F877A ha effettuato la sottrazione, riportato il valore del risultato all’esterno e che noi abbiamo studiato il comportamento assunto dai flag del registro STATUS, per cancellare la sottrazione appena effettuata dobbiamo nuovamente premere e rilasciare il pulsante collegato al bit 0 del registro PORTE e, successivamente, il µC ritornerà a leggere e memorizzare il valore che inseriamo nel registro PORTD nell’attesa di effettuare una nuova sottrazione.

Misure e collaudo del circuito:

Qui sotto è presente la tabella riguardante le prove effettuate per studiare il comportamento dei flag Z, DC e C del registro STATUS, cioè le loro variazioni di stato logico, dopo che è stata effettuata la sottrazione (ricordatevi che, all’inizio del programma e dopo tutte varie cancellature delle sottrazioni effettuate per effettuarne delle successive, il bit 2 dello STATUS, cioè il flag Z, è a stato logico “0”, mentre i bit 1 e 0, rispettivamente flag DC e C dello STATUS, sono a stato logico “1”, siccome il borrow funziona in logica negativa):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N° caso | Sistema numerico dei numeri utilizzati nella sottrazione | Valore in PORTD | - | | Valore in PORTC | Valore in PORTB (il risultato della sottrazione) | Z | DC | C |
| 1° | Decimale | 170 | | 85 | | 85 | 0 | 1 | 1 |
| Binario | 10101010 | | 01010101 | | 01010101 |
| Esadecimale | AA | | 55 | | 55 |
| 2° | Decimale | 37 | | 22 | | 15 | 0 | 0 | 1 |
| Binario | 00100101 | | 00010110 | | 00001111 |
| Esadecimale | 25 | | 16 | | 0F |
| 3° | Decimale | 159 | | 200 | | 215 | 0 | 1 | 0 |
| Binario | 10011111 | | 11001000 | | 11010111 |
| Esadecimale | 9F | | C8 | | D7 |
| 4° | Decimale | 69 | | 220 | | 105 | 0 | 0 | 0 |
| Binario | 01000101 | | 11011100 | | 01101001 |
| Esadecimale | 45 | | DC | | 69 |
| 5° | Decimale | 75 | | 75 | | 0 | 1 | 1 | 1 |
| Binario | 01001011 | | 01001011 | | 00000000 |
| Esadecimale | 4B | | 4B | | 00 |
| 6° | Decimale | 75 | | 76 | | 255 | 0 | 0 | 0 |
| Binario | 01001011 | | 01001100 | | 11111111 |
| Esadecimale | 4B | | 4C | | FF |
| 7° | Decimale | 75 | | 77 | | 254 | 0 | 0 | 0 |
| Binario | 01001011 | | 01001101 | | 11111110 |
| Esadecimale | 4B | | 4D | | FE |
| 8° | Decimale | 14 | | 255 | | 15 | 0 | 0 | 0 |
| Binario | 00001110 | | 11111111 | | 00001111 |
| Esadecimale | 0E | | FF | | 0F |
| 9° | Decimale | 15 | | 255 | | 16 | 0 | 1 | 0 |
| Binario | 00001111 | | 11111111 | | 00010000 |
| Esadecimale | 0F | | FF | | 10 |
| 10° | Decimale | 30 | | 10 | | 20 | 0 | 1 | 1 |
| Binario | 00011110 | | 00001010 | | 00010100 |
| Esadecimale | 1E | | 0A | | 14 |
| 11° | Decimale | 30 | | 20 | | 10 | 0 | 1 | 1 |
| Binario | 00011110 | | 00010100 | | 00001010 |
| Esadecimale | 1E | | 14 | | 0A |
| 12° | Decimale | 30 | | 30 | | 0 | 1 | 1 | 1 |
| Binario | 00011110 | | 00011110 | | 00000000 |
| Esadecimale | 1E | | 1E | | 00 |
| 13° | Decimale | 30 | | 40 | | 246 | 0 | 1 | 0 |
| Binario | 00011110 | | 00101000 | | 11110110 |
| Esadecimale | 1E | | 28 | | F6 |
| 14° | Decimale | 30 | | 15 | | 15 | 0 | 0 | 1 |
| Binario | 00011110 | | 00001111 | | 00001111 |
| Esadecimale | 1E | | 0F | | 0F |
| 15° | Decimale | 31 | | 16 | | 15 | 0 | 1 | 1 |
| Binario | 00011111 | | 00010000 | | 00001111 |
| Esadecimale | 1F | | 10 | | 0F |
| 16° | Decimale | 29 | | 14 | | 15 | 0 | 0 | 1 |
| Binario | 00011101 | | 00001110 | | 00001111 |
| Esadecimale | 1D | | 0E | | 0F |
| 17° | Decimale | 29 | | 15 | | 14 | 0 | 0 | 1 |
| Binario | 00011101 | | 00001111 | | 00001110 |
| Esadecimale | 1D | | 0F | | 0E |
| 18° | Decimale | 29 | | 16 | | 13 | 0 | 1 | 1 |
| Binario | 00011101 | | 00010000 | | 00001101 |
| Esadecimale | 1D | | 10 | | 0D |
| 19° | Decimale | 32 | | 17 | | 15 | 0 | 0 | 1 |
| Binario | 00100000 | | 00010001 | | 00001111 |
| Esadecimale | 20 | | 11 | | 0F |
| 20° | Decimale | 31 | | 17 | | 14 | 0 | 1 | 1 |
| Binario | 00011111 | | 00010001 | | 00001110 |
| Esadecimale | 1F | | 11 | | 0E |
| 21° | Decimale | 31 | | 18 | | 13 | 0 | 1 | 1 |
| Binario | 00011111 | | 00010010 | | 00001101 |
| Esadecimale | 1F | | 12 | | 0D |
| 22° | Decimale | 32 | | 18 | | 14 | 0 | 0 | 1 |
| Binario | 00100000 | | 00010010 | | 00001110 |
| Esadecimale | 20 | | 12 | | 0E |
| 23° | Decimale | 32 | | 20 | | 12 | 0 | 0 | 1 |
| Binario | 00100000 | | 00010100 | | 00001100 |
| Esadecimale | 20 | | 11 | | 0C |
| 24° | Decimale | 31 | | 20 | | 11 | 0 | 1 | 1 |
| Binario | 00011111 | | 00010100 | | 00001011 |
| Esadecimale | 1F | | 14 | | 0B |

Guardando i risultati delle prove che abbiamo effettuato, notiamo che vi sono alcuni casi in cui il funzionamento dei flag non è corrispondente a quanto ci aspetteremmo. Per esempio:

* Il flag Z si setta ad “1” logico quando il risultato di ogni operazione algebrica o di un’istruzione è uguale a zero e, perciò, per settarlo il byte deve avere tutti i bit a “0” logico;
* Il carry, siccome i calcoli che compie il µC sono svolti con numeri del sistema numerico binario, si setta ad “1” logico quando avviene un prestito, teorico, tra il nono bit (che non abbiamo in nessun registro) e l’ottavo del minuendo. Questo avviene, da quanto abbiamo verificato tramite le prove, quando il minuendo (1° fattore della sottrazione) ha valore minore rispetto al sottraendo (il 2° fattore della sottrazione) e, cioè, quando riotteniamo, per esempio, il valore decimale 255 dalla sottrazione o comunque, risultati della sottrazione di valore maggiore rispetto al minuendo (vedi 3°, 4°, 6°, 7°, 8°, 9° e 13° caso);
* Il digit carry, siccome i calcoli svolti dal PIC sono compiuti con il sistema numerico binario, si setta a stato logico “1” quando, nell’effettuare la sottrazione, avviene il prestito tra il quinto ed il quarto bit del minuendo (considerando i valori nel sistema numerico binario). Prendiamo come esempi il caso 8 ed il caso 15:

guardando la tabella, notiamo nel caso 8 che, dopo che il PIC ha effettuato la sottrazione, il flag DC del registro STATUS si è settato a “0” logico, segno che è avvenuto il prestito tra il quinto ed il quarto bit del byte del minuendo, perché nella sottrazione tra 14 (in binario: 00001110) e 255 (in binario:11111111) il bit 3 del minuendo (dopo alcuni calcoli) è a “0” logico, mentre il bit 3 del sottraendo è ad “1” logico e, perciò, per completare la sottrazione, il bit 3 del minuendo chiede il prestito al bit 4 del minuendo e, in questo caso, il risultato è uguale al valore massimo (in decimale) del nibble dei bit meno significativi (15). Il caso 15, invece, ottiene lo stesso risultato (decimale) del caso 8 ma il flag DC, a differenza del caso che abbiamo spiegato in precedenza, non si setta (sempre da solo) a “0” logico ma rimane a stato logico “1” perché, nonostante il valore rientri nel range da 15d a 0d, non avviene il prestito tra il bit 4 ed il bit 3 del minuendo nella sottrazione nel sistema numerico binario. Bisogna ragionare, come anche per la somma (ma al contrario), col sistema numerico binario, perché quello decimale può trarre in inganno.

Osservazioni:

Abbiamo osservato il reale comportamento dei flag Z, DC e C del registro STATUS e compreso qual sistema numerico dobbiamo considerare per capire il comportamento dei flag dopo che è stata effettuata una sottrazione, cioè considerare il sistema numerico binario.