CLASSE 2 CMP/ADD

martedì 22 novembre 2022 08:35

ISTRUZIONI LOGICO ARITMETICHE

Tipo	Mnemonico	Operandi	O S Z P C	Descrizione
0	add	B, E	⇕ ⇕ ⇕ ⇕ ⇕	Memorizza in E il risultato di E + B
1	sub	B, E	⇕ ⇕ ⇕ ⇕ ⇕	Memorizza in E il risultato di E - B
2	adc	B, E	⇕ ⇕ ⇕ ⇕ ⇕	Memorizza in D il risultato di E + B + CF
3	sbb	B, E	⇕ ⇕ ⇕ ⇕ ⇕	Memorizza in D il risultato di E - (B + neg(CF))
4	cmp	B, E	⇕ ⇕ ⇕ ⇕ ⇕	Confronta i valori di B ed E calcolando E - B, il risultato viene poi scartato
5	test	B, E	⇕ ⇕ ⇕ ⇕ ⇕	Calcola l'and logico bit a bit di B ed E, il risultato viene poi scartato
6	neg	E	⇕ ⇕ ⇕ ⇕ ⇕	Rimpiazza il valore di E con il suo complemento a 2
7	and	B, E	0 ⇕ ⇕ ⇕ 0	Memorizza in E il risultato dell'and bit a bit tra B ed E
8	or	B, E	0 ⇕ ⇕ ⇕ 0	Memorizza in E il risultato dell'or bit a bit tra B ed E
9	xor	B, E	0 ⇕ ⇕ ⇕ 0	Memorizza in E il risultato dello xor bit a bit tra B edE
10	not	E	0 ⇕ ⇕ ⇕ 0	Rimpiazza il valore di E con il suo complemento a uno
11	bt	K, E	- - - - ⇕	Imposta CF al valore del K-simo bit di E (bit testing)

VERIFICA SE IL BIT K-ESIMO VALE 0 O 1.

SONO OPERAZIONI CHE MODIFICANO IL REGISTRO DEI FLAGS, IN FUNZIONE DEL RISULTATO DELL'OPERAZIONE;

ADC = NEL PYTHON, CHE È UN LINGUAGGIO INTERPRETATO USA LA PRECISIONE ARBITRARIA. SE IL NUMERO RAPPRESENTATO CRESCE, UTILIZZERÀ PIÙ CIFRE.

IL MIO PROCESSORE HA REGISTRI A 64 BIT COME FACCIO A RAPPRESENTARE UN INTERO A 128 BIT?

USO DUE REGISTRI.

MA ALU NON È IN GRADO DI FARE LA SOMMA A 128 BIT.

DIVIDO I MIEI 2 OPERANDI A 64 BIT E SOMMO DUE A DUE.

PROPAGO IL RISULTATO DI UN BIT DI OVERFLOW IN UNA SOMMA SUCCESSIVA.

LECITO: SI PUÒ FARE!

3 ADC PER UNA SOMMA A 256 BIT

CMP

X = 0

X - 0 = 0

DESTINAZIONE SORGENTE RISULTATO

ESISTE UN BIT DI STATO CHE MI DICE SE IL RISULTATO DELLA MIA OPERAZIONE È ZERO? ZF.

↳ VALE 1 SE IL RISULTATO DI UNA SOTTRAZIONE VALE 0.

Un'operazione di uguaglianza la rappresento come una sottrazione!

$a = b$?

$a - b = 0$? → SE SÌ, SONO uguali!

CMP fa una sottrazione. Non aggiorna il contenuto del Reg. destinazione ma aggiorna ZF nei BIT di flags.

↳ sarà necessario INTERAGIRE quando scriveremo codice Assembly!