**Aritmetické instrukce**

Tento dokument je pojat jako komentář k datasheetu fy MICROCHIP. Datasheet má název 33EV32.pdf , jeho verse je DS70005144D. Versi najdete dole na stránce v pdf dokumentu. Stažený dokument máte také v <http://ozeas.sdb.cz/panska/mikroproc/33EV32/pdf/> Pokud si stáhnete nějakou jinou versi, nebudou vám souhlasit stránky a čísla kapitol. Při studiu této kapitoly očekáváme, že znáte všechno z kapitoly „Přesuny“.

Prototypem aritmetických instrukcí je instrukce add

**Add scitanec1 , scitanec2 , soucet**

Instrukce sečte scitanec1 se scitanec2 a výsledek uloží do soucet. Vidíme, že logika instrukce je stejná jako u mov, tedy místo, do kterého ukládáme výsledek, je nejvíce napravo. Lze použít všechny adresové módy, které známe z mov , například

add W1 , W2 , W3 ; sečte obsah W1 a W2 a výsledek uloží do W3

add W3 , W3, W3 ; zdvojnásobí obsah registru W3

add W3, W1, [W4] ; výsledek uloži do bunky, jejíž adresa je v reg. W4

samozřejmě add umi preinkrementaci, postinkrementaci, predekrementaci, postdekrementaci

Podíváme se do datasheetu na stranu 332 , tam máme instrukce, zkusíme si další možné módy instrukce add

Dále si pusťte program arit01.s v simulátoru a sledujte výklad i tam.

**Status register**

S aritmetickými instrukcemi je bytostně spojen stavový registr – Status register SR . V datasheetu ho máme na straně 24 . Stavový registr nás informuje o výsledku provedené operace.

Má mnoho bitů, budou nás zatím zajímat jen následující:

|  |  |  |
| --- | --- | --- |
| Číslo bitu | Označení | Význam |
| 0 | C | Carry – přetečení |
| 1 | Z | Zero – výsledek operace je 0 |
| 2 | OV | ALU Overflow bit |
| 3 | N | Negative – výsledek je záporný |
| 8 | DC | Poloviční přenos |

Bit C se nastaví do 1 , pokud se výsledek nevejde do 16ti bitů. Např. když sčítáme 0xffff + 5, vyjde 4 , a bit C má hodnotu 1 (převeďte si do binárního kódu a sečtěte na papíře, máme jenom 16 bitů)

Obdobný význam má bit DC – přenos mezi dolními a horními osmi bity. Zkuste sečíst 0x00ff + 0x0002 – dojde k přenosu. A tento přenos nastaví do 1 bit DC.

Bit Z se nastaví do 1 , když výsledek předcházející operace byl 0 . Tohle je trochu matoucí a špatně se to pamatuje, tak ještě jednou: Bit Z se nastaví do 1 , když výsledek předcházející operace byl 0 . Můžete si to pamatovat takto: “Je to zero”

**Zobrazení záporných čísel - druhý doplněk**

Náš procesor je šestnáctibitový. To znamená, že může zobrazit celkem 65536 různých čísel. Na programátorovi záleží, jak on (programátor) čísla interpretuje. **Procesoru je to úplně jedno !**

Programátor si řekne, že bude pracovat pouze s kladnými čísly. Potom má k dispozici čísla od 0 do 65535.

Nebo si řekne, že chce mít čísla kladná i záporná. Potom má k dispozici kladná čísla od 1 do 32767 a záporná čísla od -1 do -32768. Záporných čísel je o 1 víc, protože musíme zobrazit i číslo 0.

O tom, zda je číslo kladné nebo záporné, rozhoduje 15. Bit. Ten může být interpretován jako znaménko. (pokud se programátor rozhodne, že chce pracovat i se zápornými čísly)

Z kladného čísla vyrobíme záporné pomocí druhého doplňku. Postup je jednoduchý:

1. Uděláme inversi všech bitů
2. Přičteme 1

Přitom nezapomeneme, že máme 16 bitů, takže číslo vždy doplníme zleva nulami na 16 bitů.

Příklad: chceme udělat druhý doplněk k číslu 26 , tedy získat číslo -26

0000 0000 0001 1010 číslo 26 binárně, doplněno na 16 bitů

1111 1111 1110 0101 invertujeme bity

0000 0000 0000 0001 jednička

1111 1111 1110 0110 a sčítáme - výsledek je 0xffe6

Protože je poněkud matoucí, když máme postupně všechna čísla nad sebou, tady dále je ještě jednou to, co máme sčítat, a výsledek:

1111 1111 1110 0101 invertované bity

0000 0000 0000 0001 jednička

----------------------------------------------------

1111 1111 1110 0110 a sčítáme - výsledek je 0xffe6

Náš překladač je šikovný, schroustá i číslo -26 desítkově - podívejte se opět do arit01.s

Ještě jednou zdůrazňuji: procesoru je úplně jedno, jak já, programátor, chápu bity v číslech. Procesor s nimi vždy udělá totéž a nastaví bity STATUS registru. Bit N – znaménko – prostě kopíruje 15. bit výsledku.

**ALU Overflow bit**

Jako největší šamanství vypadá bit OV, ale opět je to hračka.

Udělejte si součet ( -30123 ) + ( -28554).

Ano, chceme sečíst dvě záporná čísla.

Čísla převeďte do binárního tvaru, udělejte si od obou druhý doplněk, a sečtěte je. Nezapomeneme, že máme jenom 16 bitů !

Co vidíme ? Vyšlo číslo kladné.

Ale právem bychom očekávali, že výsledek bude záporný ( -58667).

Ale tohle číslo se nám do procesoru nevejde. A právě o této situaci nás informuje bit OV.

Stejně tak se nastaví, když sčítáme dvě velká kladná čísla a vyjde číslo záporné.

Další instrukce je instrukce pro odčítání

**Sub mensenec , mensitel , rozdil**

Význam je snad jasný, opět je možno použít všechny módy adresování jako u add

Instrukci sub provádí procesor tak, že přičte druhý doplněk menšitele

Dále se k aritmetickým instrukcím počítají instrukce inc, dec, inc2, dec2 – jsou jasně vysvětleny ve file arit01.s . Přičítají a odčítají 1 nebo 2

V tabulce instrukcí na straně 331datasheetu je sloupec “Status Flags Affected” . Tam je uvedeno, které bity STATUS registru daná instrukce mění. Ostatní bity zůstávají nezměněny. **Nezměněny!** To znamená, že zůstanou ve stavu, v jakém byly před provedením instrukce. Nenulují se ! podívejte se na to !