Semestrální práce z předmětu 36NLP

Martin Tůma, 3/50 (paralelka 109)

15.11.2004

Obsah

1 Zadání

Součet modulo 2 (funkce „XOR“) dvou stejně dlouhých operandů se uloží na místo prvního z nich. Délka operandů ve slabikách (1 slabika = 1B = 8b) je zapsána v instrukci za operačním znakem jako přímý operand. Počáteční adresa prvního operandu je uložena v registru D, počáteční adresa druhého operandu v registru S. Příznaky CF, OF, SF, ZF a AF musí být nastaveny definovaným způsobem, a to (pokud možno) konzistentně s ostatnímy instrukcemi. Obsah registrů D a S může být po provedení operace změněn definovaným způsobem. Obsah registru W nemusí být po provedení operace definován.

2 Řešení

2.1 Popis instrukce

Instrukce MXOR (Memory XOR), operační znak 0Ch, provádí funkci XOR dvou operandů uložených v paměti, přičemž výsledek ukládá na místo prvního operandu. Počáteční adresa prvního operandu se bere z registru D, počáteční adresa druhého operandu z registru S. Délka obou operandů ve slabikách je uložena v instrukci za operačním znakem jako přímý operand. Instrukce nastavujue příznaky stejně jako operace XOR, konkrétně tedy takto:

$$\text{CF} = 0,\ \text{SF} = \sigma,\ \text{ZF} = \zeta,\ \text{OF} = 0,\ \text{AF} = \nu$$

kde σ = MSB[1] výsledku, ν = MSB druhého operandu a $\zeta = 1$ pokud je výsledek 0, jinak je $\zeta = 0$.

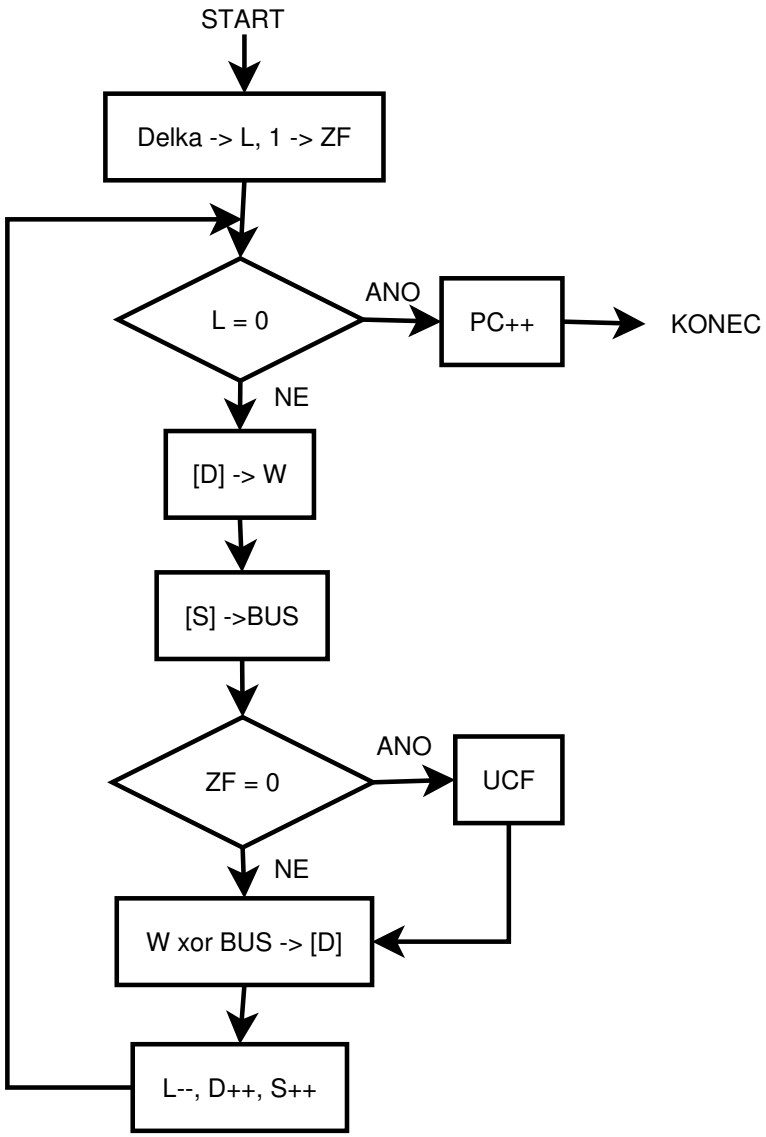
Po provedení instrukce jsou registry procesoru nastaveny následujícím způsobem: V registrech D a S je adresa bytů následujících za použitými operandy (tzn. $D = D + d$, $S = S + d$, kde d je délka operandů), registr L je vynulován a obsah registru W není definován.

2.2 Použitý algoritmus

Použitý algoritmus je velmi prostý. Postupně je aplikována funkce XOR na jednotlivé byty operandů, od nejnižších bytů po nejvyšší a tyto částečné výsledky jsou rovnou zapisovány na patřičné místo do paměti. Pokud je v průběhu „xorování“ libovolného bytu nastaven ZF na 0, je při zpracování dalších bytů použit signál UCF, který zamezí další změně ZF. Tím docílíme toho, že po skončení „xorování“ všech bytů bude ZF nastaven na jedna pouze pokud každý jednotlivý byte byl nulový.

[1] MSB - Most Significant Bit (nejvíce významný bit)

2.3 Vývojový diagram



2.4 Implementace

```
;-------------------------------------------
; VLASTNI INSTRUKCE (0C)  - MXOR
;-------------------------------------------

MXOR:       OEPC MRD OEAB ECINL          ; delka -> DIL
            OEINZE PEL, 0, 0,            ; DIL -> L
            ECW ECF, 0, 0,               ; 0->W => 1->ZF


TEST:       , 0 , 14 , CYKLUS            ; L=0 ?
CYKLUS:     {
            , 0 , 0 , DALSI_BYTE         ; L>0 => XOR dalsi byte
            ECPC, 0 , 0 , INTCHECK       ; L=0 => PC++, konec
            }


DALSI_BYTE: {
            OES OEAB MRD ECINH           ; [S] -> DIH
            OEINLH PET                   ; DI -> T

            OED OEAB MRD ECINH           ; [D] -> DIH
            OEINLH ECW, 2, 17, ZFTEST    ; DI -> W, ZF=0 ?
            }
ZFTEST:     {
            UCF,,, ZAPIS                 ; ZF=0 => UCF
            ,,, ZAPIS                    ; ZF=1
            }
ZAPIS:      {
            OET ECW ECF, 6, 0,           ; W XOR T -> W
            OEW ECOUTWR                  ; W -> DO
            OED OEAB LH OEWR MWR,0,0,    ; DOH -> [D]

            ECL ECD ECS, 0, 0, TEST      ; L--, D++, S++, jmp na TEST
            }
```

Ve výpisu není uvedeno přesměrování hlavního mikroprogramu na vlastní instrukci. Toto se provede nastavením skoku na návěští MXOR v poslední rozhodovací úrovni pro vyhodnocení operačního znaku 0Ch.

2.5 Výsledky ladění

Vstup:

	PC	[PC+1]	D	S	[D…D+d]	[S…S+d]
1	00h	02h	0004h	0008h	BBAAh	3412h
2	00h	03h	0004h	0008h	AABBCCh	AABBCCh
3	00h	03h	0004h	0008h	AABBCCh	AADDCCh
4	00h	00h	0004h	0008h	AAh	BBh

Výstup:

	PC	D	S	[D…D+d]	CF	ZF	OF	SF	AF
1	02h	0006h	000Ah	8FB8h	0	0	0	1	0
2	02h	0007h	000Bh	000000h	0	1	0	0	1
3	02h	0007h	000Bh	006600h	0	0	0	0	1
4	02h	0004h	0008h	AAh	0	1	0	0	0

Z výsledků lze usuzovat, že instrukce provádí korektně všechny operace, které má. Pouze ve specifickém případě (4), kdy je počet slabik operandů roven 0 je diskutabilní nastavení příznaku ZF na 1. (Jak je definována funkce XOR dvou prázdných množin?)

2.6 Kvantitativní charakteristiky

Jak lze již ze zadání odhadnout, je časová složitost instrukce závislá na počtu slabik d. Složitost samotné instrukce v mikroinstrukcích:

$$T = 3 + 2(d + 1) + 9d = 5 + 11d$$

Pokud uvážíme i dekódování operačního znaku (0Ch), je celková složitost instrukce dána výrazem:

$$T_c = T_{dekod} + T = 6 + (5 + 11d) = 11(d + 1)$$

3 Závěr

Navržená instrukce plně vyhovuje zadání, navržené řešení je plně funkční a pro malé délky operandů i poměrně efektivní. V případě, že by byla instrukce primárně určena pro „xorování“ velkých oblastí paměti, bylo by vhodnější instrukci koncipovat tak, aby funkci XOR prováděla po slovech s případným samotným vyhodnocením posledního lichého bytu. Při současném řešení je vlastně půlka výpočetního výkonu ALU nevyužita.

K samotné semestrální práci se lze vyjádřit velice kladně. Materiály k vypracování jsou dobře dostupné a dostačující, problém je pouze se v nich naučit orientovat. (Což ale

u „skutečné“ dokumentace je obdobné) Zadání není nijak extrémně složité, vyžaduje ale perfektní pochopení vnitřní struktury procesoru DOP.