

## Detekcia CPU a jeho schopnosti

Detekcia procesora a jeho schopností je dôležitá v prípade, že potrebujete napísať programový kód výkonný na širokej škále procesorov. V prípade dnešných procesorov je detekcia podporovaných inštrukčných sad viac menej zbytočná, pretože s najväčšou pravdepodobnosťou podporujú všetky, stále však existuje obrovské množstvo majiteľov počítačov osadených procesormi, u ktorých to tak nie je (napr. prvé revízie AMD Athlonov 64 a ich deriváty).

A na čo je to vlastne dobré? Ak sa assembleru (len pre jednoduchosť, správne som mal napísať jazyk symbolických adries) venujete trochu viac, určite poznáte výhody inštrukčných sad ako MMX a SSE. Pre tých, ktorí nechápú, o čom sa tu bude písať, bude asi treba situáciu objasniť...

Väčšina inštrukčných sad MMX a SSE môže programátor pracovať súčasne s viac ako jednou celočíselnou hodnotou (MMX, SSE) alebo s viacerými hodnotami s pohyblivou desatinnou čiarkou (SSE). Registre MMX sú dlhé 64 bitov, registre SSE majú dĺžku 128 bitov.

Inštrukčná sada MMX sa po prvý krát objavila v Pentiu MMX. Programy optimalizované pre MMX vykazovali veľmi slušný nárast výkonu oproti dovtedy používaným (kľudne aj o >50%). Vyšší výkon bol daný už spomínanou väčšou šírkou MMX registrov 64 bitov sa totiž zmestia dve 32-bitové, či štyri 16-bitové hodnoty a procesor ich môže spracovať naraz. Napriek tomu, čo skratka MMX symbolizuje, bola určená len na prácu s celočíselnými údajmi a v prípade grafických operácií stála význam. To u Intelu vyriešili až o dve procesorové generácie neskôr, v procesoroch Pentium III. SSE boli určené aj na prácu s desatinnými číslami a to ich spolu s faktom, že dĺžka premenných reálneho typu je v 32-bitovom prostredí, kvôli rýchlosti, 32 bitov, priam predurčuje na grafické výpočty, v ktorých potom CPU dokáže úplne nahradiť staré dobré FPU.

Program na zistenie podporovaných inštrukčných sad je veľmi jednoduchý, napísaný v C++ a nezávislý od operačného systému. Najprv som navrhol jednoduchú štruktúru obsahujúcu hodnoty, ktoré budeme nastavovať. Nasleduje funkcia, v ktorej prebieha detekcia procesora. Viac bude uvedené v komentároch zdrojového kódu.

```
typedef struct sCPU {
    bool MMX;
    bool SSE;
    bool SSE2;
    bool SSE3;

    sCPU() {
        MMX = false;
        SSE = false;
        SSE2 = false;
        SSE3 = false;
    }
} tCPU;

tCPU getCPUinfo() {
    tCPU cpuinfo;

    _asm {
        mov eax, 1 // ak register EAX nastavíme na 1 a zavoláme
```

```

cpuid    // inštrukciu CPUID, do registru EDX sa uloží
         // informácia o podporovaných inštruktých sadách

    test edx, 06000000h // podpora SSE3 sa nachádza na 27. bite
jz __SSE2
mov [cpuinfo.SSE3], 1

__SSE2:
test edx, 04000000h // podpora SSE2 sa nachádza na 26. bite
jz __SSE
mov [cpuinfo.SSE2], 1

__SSE:
    test edx, 02000000h // podpora SSE sa nachádza na 25. bite
jz __MMX
mov [cpuinfo.SSE], 1

__MMX:
    test edx, 00800000h // podpora MMX sa nachádza na 23. bite
jz __END
mov [cpuinfo.MMX], 1

__END:
}

return cpuinfo;
}

```

Do registru EAX priradíme 1, zavoláme CPUID a tá do registru EDX uloží informáciu o podporovaných inštrukciách. Tú nám už len stačí prečítať.

Mohlo by sa stať, že niekto z vás spustí program obsahujúci túto detekciu na počítači s procesorom starším ako 80486DX4. Asi by sa zrušilo, pretože staršie procesory inštrukciu CPUID nepoznajú. Ošetrenie tohto problému je jednoduché, detekciu stačí uzavrieť do bloku try - except.

```

__try {
    _asm {
        // detekcia
        ...
    }
}
__except(EXCEPTION_EXECUTE_HANDLER) {
    if ( _exception_code() == STATUS_ILLEGAL_INSTRUCTION )
        // inštrukcia CPUID nie je podporovaná
        return cpuinfo;
    else
        // nastal nejaký iný problém
    return cpuinfo;
}

```