
Informatyka Studia Dienne
Laboratorium Architektury Komputerów

Ćwiczenie 7

Identyfikacja procesora.
Rozkaz CPUID.

Przygotowanie:
Krzysztof Tokarz

1. Cel ćwiczenia.

Celem ćwiczenia jest poznanie:

- modeli procesorów zgodnych z 8086,
- działania rozkazu identyfikacji procesora CPUID,
- konwersji liczb szesnastkowych na dziesiętne,
- konwersji liczb z postaci binarnej na tekstową.

2. Procesory zgodne z 8086.

Procesory Intel 8086 swoją popularność zawdzięczają ich wykorzystaniu w komputerach klasy PC od początku ich powstania. Popularność tych procesorów zaowocowała ich rozbudową w kolejnych konstrukcjach firmy Intel, oraz powstaniem kompatybilnych procesorów zaprojektowanych przez innych producentów. Kolejne procesory firmy Intel powstałe na bazie 8086 to 80186, 80286, 80386, i486, Pentium, Pentium Pro, Pentium II, Pentium III, Pentium 4. Każdy z następców 16-bitowego 8086 potrafi wykonywać kod dla niego napisany i dysponuje takim samym zestawem podstawowych rejestrów. Pojawiające się w kolejnych konstrukcjach (nie tylko firmy Intel) rozszerzenia listy rozkazów (MMX, SSE, SSE2, SSE3, 3DNow) wymusiły wprowadzenie identyfikacji procesora na którym jest uruchomione oprogramowanie.

3. Rozkaz CPUID.

Począwszy od procesora Pentium do listy rozkazów została wprowadzona instrukcja CPUID pozwalająca na identyfikację modelu procesora. Rozkaz CPUID wymaga podania w rejestrze EAX numeru żądanej funkcji. Zestawienie funkcji przedstawiono poniżej:

- 00000000h – zwraca w EAX najwyższy numer funkcji rozkazu CPUID akceptowany przez procesor, w rejestrach EBX:EDX:ECX ciąg znaków określających producenta procesora,
- 00000001h – zwraca w EAX sygnaturę procesora (typ, rodzina, model), w EBX identyfikator marki, w EDX mapę bitową określającą cechy procesora,
- 00000002h – zwraca w EAX:EBX:EDX:ECX informację o pamięci podręcznej (Cache),
- 00000003h – zwraca w EDX:ECX 64 bity numeru seryjnego,
- 00000004h – zwraca w EAX:EBX:EDX:ECX nazwę procesora.

Przed wywołaniem rozkazu CPUID należy sprawdzić czy jest on obsługiwany. Starsze modele, nie obsługujące tego rozkazu, nie rozpoznają go i wygenerują błąd. Rozkaz CPUID jest obsługiwany jeśli bit 21 rejestru EFLAGS da się modyfikować.

Najważniejsze dla programisty informacje zwracane są przez funkcję nr 1 w rejestrze EDX. Określone bity rejestru oznaczają możliwość wykonania pewnych instrukcji lub grup instrukcji. Zestawienie najważniejszych bitów przedstawiono w tabeli:

Bit	Nazwa	Opis
0	FPU	Procesor obsługuje instrukcje koprocessora numerycznego 387
4	TSC	Procesor obsługuje odczyt liczników czasu RDTSC
8	CX8	Procesor może wykonać porównanie i zamianę 8 bajtów
11	SEP	Procesor może wykonać instrukcje SYSENTER i SYSEXIT
15	CMOV	Procesor obsługuje warunkowe przenoszenie danych
19	CLFSH	Procesor może wykonać instrukcję CLFLUSH
23	MMX	Procesor wykonuje instrukcje grupy MMX i posiada rejestry MM0-MM7
25	SSE	Procesor wykonuje instrukcje grupy SSE i posiada rejestry XMM0-XMM7
26	SSE2	Procesor wykonuje instrukcje grupy SSE2

Przykład sprawdzenia czy rozkaz CPUID jest wbudowany w procesor i jego wywołania przedstawiono poniżej.

```
;--- sprawdzenie czy dziala rozkaz cpuid -----  
  
    pushfd          ;EFLAGS do EAX poprzez stos  
    pop  EAX  
    bts  EAX, 21    ;ustawienie bitu 21  
    push EAX        ;EAX do EFLAGS poprzez stos  
    popfd  
    pushfd          ;jeszcze raz EFLAGS do EAX  
    pop  EAX  
    bt   EAX, 21    ;sprawdzenie czy bit 21 dal sie ustawic  
    jnc  brak_CPUID  
  
;--- cpuid dziala mozna wywolac funkcje "0" -----  
  
    xor  EAX, EAX   ;wyzerowanie EAX  
    cpuid          ;wykonanie funkcji numer "0"  
    ...           ;obsługa wyników funkcji  
  
    mov  EAX, 1  
    cpuid          ;wykonanie funkcji numer "1"  
    ...           ;obsługa wyników funkcji  
  
;--- cpuid nie dziala -----  
brak_CPUID:  
    ...
```

4. Konwersja liczby binarnej 0-15 na liczbę szesnastkową.

Funkcja numer 1 rozkazu CPUID zwraca sygnaturę procesora. Aby ją wypisać na ekran należy zamienić wartości binarne na odpowiadające im kody ASCII znaków do wyświetlenia. Można to zrobić wykorzystując instrukcję XLATB jak w poniższej procedurze, która zamienia liczbę w AL na odpowiadającą jej cyfrę szesnastkową przedstawioną w kodzie ASCII.

```
HexStr    DB "0123456789ABCDEF"  
BinToHex:  
    push BX  
    mov  BX, OFFSET HexStr  
    and  AL, 0Fh  
    xlatb  
    pop  BX  
    ret
```