2. Команда CPUID та її використання. Приклади

Ця команда призначена для ідентифікації процесора та отримання відомостей про його властивості. Для того, щоб вказати, які саме відомості потрібні, треба записати у регістр EAX деяке значення – параметр для команди. Результати роботи команди CPUID процесор записує у регістри EAX, EBX, ECX EDX. Для того, щоб правильно користуватися командою CPUID, потрібно дотримуватися певної послідовності роботи програми. Можна спочатку перевірити – а чи можливо взагалі користуватися цією командою? Для цього треба перевірити значення 21-го біту регістру EFLAGS. Проте, у цій лабораторній роботі можна вважати це зайвим.
Отримання відомостей за допомогою команди CPUID робиться у декілька кроків. Спочатку треба виконати цю команду із значенням параметру 0:

```
mov eax,0
cpuid
```

Після виконання такого коду, у регістрі EAX міститься значення, яке означає максимально можливе значення параметру, яке можна використати для отримання базових відомостей. Якщо намагатися викликати команду CPUID із параметром більше зазначеного вище максимального, то результатом будуть усі нулі (це не стосується параметрів так званих розширених відомостей). Після виконання команди CPUID із параметром 0 процесор _____також записує у регістри EBX, ECX та EDX коди символів імені процесора. Для процесорів Intel це буде "GenuineIntel", для процесорів AMD – "AuthenticAMD" тощо. Необхідно відзначити, що 12 символів записуються четвірками у такому порядку – спочатку EBX, потім EDX, останні чотири у регістрі ECX.
Наступним кроком у програмі буде виконання команди з параметром 1:

```
mov eax,1
cpuid
```

Після виконання цієї команди у регістри EAX, EBX, ECX та EDX буде записано інформацію про сімейство процесорів, модель та деякі інші відомості.
Наступним кроком у програмі буде виконання команди з параметром 2:

```
mov eax,2
cpuid
```

і так далі, наскільки можливо. Як вже вказувалося вище, максимально можливе значення параметру стає відомим після виконання команди з параметром 0. Це для базового набору відомостей. Проте, є ще так звані, розширені відомості, які відповідають функціям на основі команди CPUID з параметром 80000000h і більше. Для того, щоб дізнатися максимальне значення для параметру розширених функцій, потрібно виконати:

```
mov eax,80000000h
cpuid
```

У результаті виконання цього у регістрі EAX буде деяке значення, наприклад, 80000008h. Можна послідовно виконувати команди CPUID із значеннями параметрів від 80000001h до 80000008h для отримання відповідних відомостей. Наприклад, після виконання коду:

```
mov eax,80000008h
cpuid
```

у регістрі EAX буде інформація про максимальну можливу розрядність адрес пам’яті даного процесора.
Вичерпна інформація щодо CPUID міститься у документі "Intel® 64 and IA-32 Architectures Software Developer’s Manual Volume 2A: Instruction Set Reference", який можна завантажити через інтернет по адресі http://www.intel.com.
У підсумку, виконання ланцюжка команд CPUID можна відобразити так:

```
;---базові функції---
mov eax,0 ;початок
cpuid
. . . ;зберігання значень EAX,EBX,ECX,EDX
mov eax,1
cpuid
. . . ;зберігання значень EAX,EBX,ECX,EDX
. . . ;якщо можливо, то інші функції для EAX>1
;---розширені функції---
mov eax,80000000h
cpuid
. . . ;зберігання значень EAX,EBX,ECX,EDX
mov eax,80000001h ;якщо можливо
cpuid
. . . ;зберігання значень EAX,EBX,ECX,EDX
. . . ;якщо можливо, то інші функції для EAX>80000001h
```

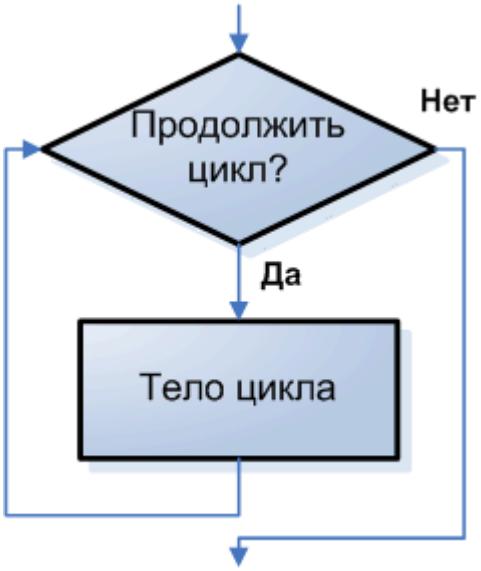
Для зберігання значень регістрів EAX, EBX, ECX та EDX можна їх записувати у масив-вектор, наприклад:

```
.data
res dd 256 dup(0)
.code
mov eax, 0
cpuid
mov dword ptr[res], eax
mov dword ptr[res+4], ebx
mov dword ptr[res+8], ecx
mov dword ptr[res+12], edx
. . . ;у подібний спосіб і для інших CPUID
```

Проте, зберігання четвірок 32-бітових значень можна запрограмувати, як здається, і по-іншому.

17. Програмування циклів з передумовою

Це цикл, у якому перевірка умови виходу з цикла виконується перед виконанням тіла циклу. На асемблері такий цикл можна запрограмувати наступним чином:




```
                mov dl, 0
cycle:          cmp dl, 9
                ja end_cycle
                inc dl
                jmp cycle
end_cycle:
```

(цикл, в кожній ітерації якого інкрементується вміст регістру dl (початкове значення - 0), поки він не дорівнюватиме 9; умова виходу з циклу – dl > 9)