Ćwiczenie nr 5

Zaawansowane techniki przekazywania parametrów

5.1 Wstęp

Podprogramy stanowią ze swej natury, w pewien sposób wyizolowaną część programu, a komunikacja z nimi odbywa się wg ściśle ustalonego protokołu, określającego formaty danych i wzajemne obowiązki programu wywołującego i wywoływanej procedury. Protokół ten nazywany jest także opisem interfejsu podprogramu.

Powyższe rozważania wskazują, że interfejs do podprogramów musi być jasno i przejrzyście zdefiniowany. W szczególności producenci oprogramowania ustalają pewne niskopoziomowe protokoły wywoływania podprogramów, przeznaczone dla wytwarzanych przez nich translatorów języków programowania. Znajomość mechanizmów wywoływania podprogramów i przekazywania parametrów stanowi podstawę do programowania mieszanego, co stanowi treść ćwiczenia 6.

5.2 Interfejs do podprogramu

Wspomnieliśmy wcześniej, że komunikacja z podprogramami musi odbywać się wg ściśle ustalonego protokołu, określającego formaty danych i wzajemne obowiązki programu wywołującego i wywoływanej procedury. Protokół ten nazywany jest także opisem interfejsu procedury. Interfejs między procedurą i wywołującym ją programem określony jest przez następujące elementy:

- sposób przekazania sterowania z programu do podprogramu;
- sposób przekazania sterowania z podprogramu do programu;
- sposób rejestrowania i odtwarzania stanu procesora;
- sposób przekazywania parametrów do podprogramu i przekazywania parametrów z podprogramu do programu wywołującego.

W dalszym ciągu elementy te rozpatrzymy bardziej szczegółowo.

W komputerach osobistych zbudowanych w oparciu o procesory Pentium wywołanie procedury sprowadza się do wykonania poniższych czynności:

- zapamiętanie na stosie adresu instrukcji, która będzie wykonana po zakończeniu wykonywania procedury i przekazaniu sterowania z powrotem do programu wywołującego;
- wpisanie do rejestru EIP adresu wskazującego położenie wywoływanej procedury.

Wymienione czynności realizuje instrukcja `call`. Instrukcja ta zapamiętuje na stosie informacje, które będą później potrzebne do ponownego przekazania sterowania z procedury do programu wywołującego.

Informacje zapisane na stosie pozwalają, po zakończeniu wykonywania procedury, na przekazanie sterowania z procedury do programu wywołującego – czynności te wykonuje instrukcja `ret`. Interfejs do procedury musi zawierać też ustalenia dotyczące zapamiętywania i odtwarzania rejestrów. W praktyce stosowane są dwie metody:

1. rejestracja i odtwarzanie stanu procesora należy do obowiązków programu wywołującego – oznacza to, że program wywołujący zapamiętuje tylko te rejestry, które zawierają potrzebne mu informacje, a wywołana procedura może dowolnie zmieniać zawartości wszystkich rejestrów roboczych;
2. rejestracja i odtwarzanie stanu procesora prowadzone jest wewnątrz procedury (wszystkie rejestry robocze używane przez procedurę są zapamiętywane a potem odtwarzane).

Zauważmy, że metoda (2) centralizuje zapamiętywanie rejestrów, podczas gdy metoda (1) wymaga zapamiętywania i odtwarzania rejestrów, odpowiednio, przed i po każdym wywołaniu podprogramu. Jeśli procedura wywoływana jest wielokrotnie, to kod programu w przypadku (1) będzie wyraźnie dłuższy. Z kolei w metodzie (2) zapamiętywane i odtwarzane mogą być także i te rejestry, których zawartość nie ma znaczenia dla programu wywołującego. Wydaje się, że metoda (2) stosowana jest częściej.

Ostatnim elementem składającym się na opis interfejsu podprogramu są zasady przekazywania parametrów do i z procedury. O ile poprzednio omówione elementy są w zasadzie stałe dla różnych systemów programowania komputerów, to sposób przekazywania parametrów różni się dość znacznie w poszczególnych systemach. Używane są dwa podstawowe sposoby przekazywania parametrów:

- przekazywanie przez wartość (ang. *call by value*), co oznacza, że do podprogramu przekazywana jest bezpośrednio wartość parametru, którym może być liczba, tablica liczb, łańcuch znaków, itp.;
- przekazywanie przez adres (ang. *call by location*), co oznacza, że do podprogramu przekazywany jest adres lokacji pamięci, w której znajduje się wartość podawana do podprogramu.

Przekazywanie adresu zmiennej zamiast jej wartości nieznacznie wydłuża kod procedury – trzeba bowiem wprowadzić dodatkowe rozkazy, które wyznaczą wartość zmiennej w oparciu o podany adres. Jednak taki sposób przekazywania ma sporo zalet, szczególnie gdy parametrem jest tablica – można wówczas zamiast przekazywania wartości poszczególnych elementów tablicy podać tylko adres jej pierwszego elementu. Ponadto, przy przekazywaniu „przez adres" podprogram uzyskuje bezpośredni dostęp do zmiennej w programie wywołującym. Pozwala to m.in. na wpisanie wyniku uzyskanego w trakcie wykonywania podprogramu bezpośrednio do tej zmiennej.

Parametry te, zarówno w postaci jawnych wartości, jak też i adresów mogą być przekazywane do podprogramu różnymi drogami. I tak parametry podprogramów systemowych BIOSu i DOSu przekazuje się najczęściej poprzez rejestry robocze. Najbardziej rozpowszechnione jest jednak przekazywanie parametrów przez stos: program wywołujący wpisuje wymagane parametry na stos, po czym wykonuje instrukcję `call`. Znane jest także przekazywanie parametrów „przez ślad", czyli przez obszary pamięci znajdujące się bezpośrednio za instrukcją `call`. Wreszcie można przekazywać parametry przez zarezerwowane obszary pamięci o ustalonych adresach.

Dla każdej z wymienionych dróg przekazywania parametrów trzeba poczynić ustalenia dotyczące przekazywania parametrów w rejestrach roboczych, określić kto jest odpowiedzialny (podprogram czy program wywołujący) za usunięcie wykorzystanych parametrów ze stosu, i podać ewentualnie inne ograniczenia.

5.3 Technika przekazywania parametrów przez stos

Przekazywanie parametrów przez stos może być obecnie uważane za typową, powszechnie używaną metodę. Metodę tę wyjaśnimy na przykładzie. Przypuśćmy, że koszt wykonania pewnej instalacji w pomieszczeniu o wymiarach długość, szerokość, wysokość określony jest przez wyrażenie $(\text{długość} + 5) \cdot \text{szerokość} + \text{wysokość}$. Zadanie polega na ułożeniu procedury obliczającej wartość tego wyrażenia. Przyjmujemy też następujące założenia:

- procesor pracuje w trybie 32-bitowym;
- do wywoływania podprogramu zostanie użyta instrukcja `call` typu NEAR;
- podane parametry, w postaci liczb 32-bitowych bez znaku, wpisywane są na stos przed wywołaniem podprogramu;

- obliczona wartość wyrażenia (zakładamy, że również 32-bitowa) powinna zostać załadowana do rejestru EAX;
- obowiązek zdjęcia parametrów ze stosu po wykonaniu obliczeń należy do programu, który wywołał podprogram.

Nazwijmy omawiany podprogram „koszt". Jeśli przykładowe wartości parametrów będą wynosiły: długość = 17, szerokość = 5, wysokość = 7, to wywołanie podprogramu może mieć postać:

```
push  17          ;ładowanie parametru długość
push  5           ;ładowanie parametru szerokość
push  7           ;ładowanie parametru wysokość
call  koszt
```

W trakcie wykonywania podanej sekwencji rozkazów na stos ładowane będą kolejne parametry, a następnie ślad rozkazu CALL. Proces ten pokazany jest na poniższym rysunku: każdy prostokąt reprezentuje pojedynczy element stosu (4 bajty), a bajty umieszczone niżej mają niższe adresy.

długość	długość	długość	długość
	szerokość	szerokość	szerokość
		wysokość	wysokość
			EIP

Obliczenia wykonywane przez omawiany podprogram są bardzo proste, pewnej uwagi wymaga natomiast technika odczytywania parametrów ze stosu. Ponieważ wartości parametrów długość, szerokość i wysokość są "przykryte" przez wartość EIP, więc w celu uzyskania dostępu do tych parametrów należałoby najpierw zdjąć ze stosu (korzystając z instrukcji POP) i przechować zawartość EIP. Następnie można by kolejno zdejmować wartości podanych parametrów wysokość, szerokość i długość, a po wykonaniu obliczeń ponownie załadować uprzednio zapamiętane zawartość EIP.

Taka technika, chociaż pozornie nieskomplikowana, ma kilka wad. Zawartość EIP trzeba bowiem zdejmować i kłaść na stos, chociaż nie jest ona bezpośrednio używana w obliczeniach. Parametry zdjęte ze stosu należy też przechowywać, ponieważ w bardziej złożonych procedurach są one zazwyczaj wielokrotnie używane.

Ponieważ operacje pobierania parametrów procedury ze stosu wykonywane są w prawie wszystkich programach, więc konstruktorzy procesorów rodziny Pentium (i jego poprzedników) opracowali specjalny mechanizm przeznaczony do pobierania parametrów ze stosu. Przyjęto mianowicie, że modyfikacja adresowa wykonywana za pomocą rejestru EBP pociąga za sobą użycie rejestru SS jako domyślnego rejestru segmentowego. Zatem odczyt pewnej lokacji pamięci, której adres podany jest w rejestrze EBP oznacza w istocie odczyt lokacji pamięci zapisanej na stosie. Wystarczy więc tylko wpisać do rejestru EBP zawartość wskaźnika stosu ESP, by natychmiast uzyskać możliwość łatwego dostępu do zawartości wierzchołka stosu i zawartości kolejnych słów zapisanych poniżej wierzchołka (pamiętamy, że stos „rośnie" w kierunku malejących adresów, zatem lokacje bardziej odległe od wierzchołka stosu będą miały wyższe adresy).

Opisaną tu technikę dostępu do parametrów zapisanych na stosie zastosowano w niżej podanej procedurze przykładowej koszt.

```
koszt PROC
push  ebp               ;przechowanie EBP na stosie
mov   ebp, esp          ;po wykonaniu tej instrukcji rejestr EBP
                        ;będzie wskazywał wierzchołek stosu
mov   eax, [ebp]+16     ;ładowanie parametru długość
add   eax, 5            ;EAX←EAX+5
mul   dword PTR[ebp]+12 ;mnożenie przez szerokość
add   eax, [ebp]+8      ;dodanie wartości parametru wysokość
pop   ebp               ;odtworzenie rejestru EBP
ret                     ;powrót do programu wywołującego
ENDP
```

długość	[EBP]+16
szerokość	[EBP]+12
wysokość	[EBP]+8
EIP	[EBP]+4
EBP	[EBP]+0

Rozpatrzmy jeszcze raz sytuację na stosie, tym razem po wykonaniu instrukcji `push ebp`. Zauważmy, że po wykonaniu instrukcji `mov ebp, esp` wierzchołek stosu jest także wskazywany przez rejestr EBP. Zatem kolejne obiekty na stosie są wskazywane przez wyrażenia [EBP]+4, [EBP]+8, [EBP]+12, itd. Tak więc odczytanie parametru długość wymaga tylko wykonania instrukcji `mov ax, [ebp]+16`. W wyrażeniu adresowym nie potrzeba jawnie wskazywać rejestru segmentowego SS, ponieważ użycie rejestru EBP jako modyfikatora implikuje użycie SS jako domyślnego rejestru segmentowego. W analogiczny sposób można odczytać pozostałe parametry.

Obliczenia wykonywane przez podany podprogram koszt są proste i nie wymagają szerszych komentarzy. Użycie operatora PTR w jednoargumentowej instrukcji `mul dword PTR [ebp]+12` jest niezbędne, ponieważ wyrażenie [EBP]+12 wskazuje położenie pewnej lokacji nie precyzując jednak czy chodzi to o bajt czy o słowo czy podwójne słowo. Zauważmy, że 32-bitowy mnożnik implikuje 64-bitowy wynik mnożenia, umieszczany w rejestrach EDX:EAX. Ponieważ wcześniej przyjęliśmy, że wynik obliczeń będzie liczbą 32-bitową więc starszą część wyniku mnożenia (w rejestrze EDX) możemy pominąć.

Przy pobieraniu innych parametrów operator PTR nie jest używany, ponieważ długość pobieranej lokacji wynika z długości rejestru podanego w rozkazie, np. `add eax, [ebp]+8`.

Instrukcja `ret` umieszczona na końcu podprogramu zdejmuje ze stosu i ładuje rejestr EIP, przekazując tym samym sterowanie do programu wywołującego.