Raport Laboratorium 2

Organizacja i Architektura komputerów

Autor: Uladzimir Kaviaka (257276)

Cel laboratorium:

Zadanie polegało na napisaniu kalkulatora liczb zmiennoprzecinkowych, przyjęto następujące założenia:

- Wczytywanie/wypisanie realizowane za pomocą **scanf, printf**
- Liczby wpisywane/wypisywane w sposób dziesiętny
- Dodanie trybów zaokrąglania FPU dla wyniku

Opis algorytmu:

Wczytujemy dwie podane liczby, wczytujemy wybrany typ operacji, wczytujemy wybrany tryb zaokrąglania. Robimy inicjalizację jednostki FPU, następnie na podstawie trybu zaokrąglania ustawiamy słowo kontrolne dla jednostki FPU, i wykonywamy wybraną operację po czym wyświetlamy wynik

Implementacja programowa:

1. Rezerwacja pamięci pod zmienne tekstowe, zmienne operacji oraz trybu zaokrąglania oraz dla dwóch zmiennych na podstawie których będzie wykonana wybrana operacja.

```
76  .data
77      msg1: .ascii "Podaj pierwsza liczbe\n\0"
78      msg2: .ascii "Podaj druga liczbe\n\0"
79      menu: .ascii "Wybierz typ operacji\n1. Dodawanie\n2. Odejmowanie\n3. Dzielenie\n4. Mnozenie\n\0"
80      control: .ascii "Typ zaokraglania\n1. Do zera\n2. W gore\n3. W dol\n4. Do najblizszej\n\0"
81      output: .ascii "Wynik %f\n\0"
82
83      first_d: .float 0.0
84      second_d: .float 0.0
85      operation: .int 0           #zmienna dla przechowywania wybranej operacji
86      control_operation: .int 0   #zmienna dla przechowywania wybranego zaokraglania
87
88      format_d: .ascii "%f\0"     #format wypisywania float
89      format_i: .ascii "%i\0"     #format wypisywania calkowitej liczby
```

Najpierw deklarujemy zmienne dla naszego menu oraz komunikacji z użytkownikiem (linijki 77-81), następnie rezerwujemy miejsce pod dwie liczby(linijki 83-84), następnie rezerwujemy miejsce pod dwie liczby do przechowywania wybranej operacji oraz trybu zaokrąglania(linijki 85,86). Na końcu deklarujemy zmienne formatu wypisywania/wpisywania wartości które są potrzebne do do funkcji scanf/printf.

2. Inicjalizacja jednostki FPU

Inicjalizacja jednostki FPU wykonuje się za pomocą jednego rozkazu: ***finit***

3. Wyświetlanie komunikatów oraz pobranie danych wejściowych

```
 98        print_s msg1
 99        pushl $first_d
100        pushl $format_d
101        call scanf
102
103
104        print_s msg2
105        pushl $second_d
106        pushl $format_d
107        call scanf
108
109        print_s menu
110        pushl $operation
111        pushl $format_i
112        call scanf
113
114        print_s control
115        pushl $control_operation
116        pushl $format_i
117        call scanf
```

Wyświetlamy po kolei komunikaty oraz pobieramy dane wejściowe za pomocą funkcji scanf, żeby wykorzystać scanf musimy załadować argumenty tej funkcji na stos ale w odwrotnej kolejności czyli najpierw zmienna następnie format zmiennej.

4. Sprawdzenie wybranego trybu zaokrąglania

```
121        cmpb $1, control_operation
122        je cut
123        cmpb $2, control_operation
124        je up
125        cmpb $3, control_operation
126        je down
127        cmpb $4, control_operation
128        je nearest
```

Tutaj po wprowadzeniu numeru trybu zaokrąglania ten tryb jest zapisywany w zmiennej **control_operation,** po czym możemy sprawdzić co jest zapisane w tej zmiennej i następnie ustawić potrzebny tryb zaokrąglania.

5. Ustawienie trybu zaokrąglania

W FPU tryb zaokrąglania ustawiany w słowie kontrolnym, za tryb zaokrąglania odpowiadają 10 oraz 11 bit słowa kontrolnego które składa się z 16 bitów.
00 – zaokrąglenie do najbliższej
01 – zaokrąglenie w dól (-inf)
10 – zaokrąglenie do góry (+inf)
11 – obcięcie
Więc wprowadziłem cztery zmienne które będą służyć do inicjalizacji słowa kontrolnego w zależności od trybu zaokrąglania

```
1   cut1: .short 0x03F #najblizsze
2   cut2: .short 0x43F #dol
3   cut3: .short 0x83F #gora
4   cut4: .short 0xC3F #obciecie
```

Na przykładzie trybu zaokrąglenia do najbliższej bity 10 oraz 11 muszą być ustawione na zera, ale tutaj jest 0x03F co równoważne
0000 0000 0011 1111
bity zaznaczone na czerwono (0-5) odpowiada temu że ustawiamy „interrupt mask” dla FPU co pozwala generować wyjątki.

To samo jest np. dla trybu zaokrąglania w dół:
Tutaj muszą 11 oraz 10 bity ustawione jako 0 oraz 1

0000 0100 0011 1111
Analogicznie dla pozostałych trybów zaokrąglania.

6. Makra załadowania słowa kontrolnego na podstawie trybu zaokrąglania

```
70  .macro set_round_cut    #zaokraglanie przez obciecie
71      fldcw cut4
72  .endm
73
74  .macro set_round_up     #zaokraglenie do +inf
75      fldcw cut3
76  .endm
77
78  .macro set_round_down   #zaokraglenie do -inf
79      fldcw cut2
80  .endm
81
82  .macro set_round_nearest    #zaokraglenie do najblizszej
83      fldcw cut1
84  .endm
```

Do załadowania słowa kontrolnego do jednostki FPU służy rozkaz **fldcw (FPU load control word)**. Więc tutaj po prostu ładujemy słowo kontrolne do jednostki FPU

7. Makra wykonania operacji FPU na przykładzie odejmowania

```
23  .macro sub_d
24      fld first_d
25      fld second_d
26      fsubp
27      push %eax
28      push %eax
29      fstpl (%esp)
30      pushl $output
31      call printf
32      pushl $0
33  .endm
```

Najpierw ładujemy dwie zmienne do stosu FPU używając rozkaz **fld**, następnie wykonujemy rozkaz zgodny z naszą operacją czyli odejmowane co daje rozkaz FPU **fsubp.** Następnie używając **fstpl** ładujemy nasz pierwszy argument oraz drugi argument **$output** co odpowiada trybowi formatowania funkcji printf i wywołujemy printf.

Wszystkie operacje (+,-,/,*) mają ten sam algorytm wykonania za pomocą jednostki FPU. Najpierw ładujemy pierwszą zmienną używając rozkaz **fld**. Następnie wybieramy rozkaz który odpowiada operacji którą chcemy wykonać np. dodawanie **faddp** następnie ładujemy dwa parametry: zmienna oraz format do wywołania funkcji printf. Ten sam algorytm wykorzystany dla pozostałych operacji.

8. Wywołanie makro po sprawdzeniu jaki tryb zaokrąglania wybrany

```
130   cut:
131       set_round_cut
132       je rounding_set
133   up:
134       set_round_up
135       je rounding_set
136   down:
137       set_round_down
138       je rounding_set
139   nearest:
140       set_round_nearest
141       je rounding_set
```

W punkcie 4 tego raportu było sprawdzenie wybranego trybu zaokrąglania i następnie wykonujemy skok do określonego trybu, tutaj podany kod który ilustruje że po wybranym trybie zaokrąglania wywołujemy funkcję która załaduje potrzebne słowo sterujące do jednostki FPU i przechodzimy dalej.

9. Sprawdzenie wybranej operacji

```
143   rounding_set:
144       cmpb $1, operation
145       je add
146       cmpb $2, operation
147       je sub
148       cmpb $3, operation
149       je div
150       cmpb $4, operation
151       je mult
152   add:
153       add_d
154       jmp end
155   sub:
156       sub_d
157       jmp end
158   div:
159       div_d
160       jmp end
161   mult:
162       mult_d
163       jmp end
```

Tutaj znów zwykłe sprawdzenie jaką operację wybraliśmy, następnie przechodzimy do potrzebnej metki i wykonujemy operacje opisane makrami w punkcie 7 tego raportu i kończymy działanie kalkulatora.

Wnioski: Laboratorium pozwoliło zrozumieć w jaki sposób korzystać z jednostki FPU która pozwala robić obliczenia na liczbach zmiennoprzecinkowych, zapoznałem się również ze składnią słowa kontrolnego jednostki FPU, zrozumiałem jak wykonywać operację za pomocą rozkazów FPU. Również nauczyłem się pisać makra co pozwala redukować ilość kodu.