08.04.2020

Michał Dunat, 248862

Oddano: 09.04.2020

Ocena:

Operacje dodawania, odejmowania, mnożenia i dzielenia na liczbach typu floating point o pojedynczej i podwójnej precyzji.

Sprawozdanie z laboratorium „Organizacja i Architektura Komputerów”

Rok akad. 2019/2020, kierunek INF

Prowadzący: mgr inż. Tomasz Serafin

Spis treści

Przebieg pracy nad programem

Na początku pracy zapoznano się z x87 FPU – jednostką zmiennoprzecinkową procesora, a dokładniej z jej dokumentacją i materiałami umieszczonymi w Internecie. Idea operacji zmiennoprzecinkowych znacząco różnią się od stałoprzecinkowych. Po skonstruowaniu subrutyn odpowiedzialnych za poszczególne operacje pod uwagę wzięto precyzję i tryby zaokrągleń – do tego należało modyfikować słowo kontrolne. Wiedzę jak ma wyglądać słowo kontrolne dla danej precyzji i trybu zaokrąglania zaczerpnięto z dokumentacji Intela i tabel tam zamieszczonych. Chcąc kontrolować rejestry wykorzystywane podczas operacji zmiennoprzecinkowych należało użyć rozszerzonej komendy debuggera „gdb" – „info all-r". Przedostatnią częścią pracy nad programem było wygenerowanie wyjątków. Na samym końcu kod uzupełniono o szczegółowe komentarze (które są widoczne tylko w plikach z kodem).

Napotkane problemy

Jedynym poważniejszym problemem przy realizacji tego zadania był wybór precyzji obliczeń i trybu zaokrąglania. Po dokładniejszym przeczytaniu dokumentacji i materiałów umieszonych w Internecie na temat FPU – po „odkryciu" słowa sterującego sprawa się nieco rozjaśniła, lecz dostęp i modyfikacja słowa sterującego zajęło więcej czasu niż może na to wskazywać ilość kodu potrzebna do realizacji tego zadania (3 linijki).

Kluczowe fragmenty kodu

Liczby na których będą wykonywane operacje i słowo kontrolne umieszczono w pamięci:

```
.data
firstFloat: .float 13.45
secondFloat: .float 3.27
firstDouble: .double -1.5432
secondDobule: .double 0.0
FPUControlWord: .short 0x037f
```

Słowo kontrolne jednostki zmiennoprzecinkowej ma 16 bitów - bity 8 i 9 kontrolują precyzje, a 10 i 11 tryb zaokrąglania. Dla precyzji:

- 00 - pojedyncza precyzja, 10 - podwójna precyzja, 11 - rozszerzona podwójna precyzja

Dla zaokrąglania:

- 00 - do najbliższej parzystej, 01 - w dół do minus nieskończoności, 10 - do góry do plus nieskończoności, 11 - w kierunku zera

Informacje zaczerpnięte z dokumentacji Intela. Chcąc przeprowadzać operacje w podwójnej precyzji, w trybie zaokrąglania w dół do minus nieskończoności słowo kontrolne wygląda następująco: 0x067f. Ustawianie wybranego słowa kontrolnego:

```
finit
fldcw FPUControlWord
```

Na początku należy zainicjalizować jednostkę FPU, a następnie załadować słowo sterujące z pamięci. Schemat działania programu:

```
jmp floatAdd

floatAdd:
flds firstFloat
fadds secondFloat
jmp exit
```

Na samym początku skaczemy do subrutyny odpowiedzialnej za wykonanie interesującego nas działania. Każda z subrutyn zbudowana jest tak samo, na początku ładowana jest liczba do rejestru zmiennoprzecinkowego „st(0)", a następnie za pomocą drugiej instrukcji przeprowadza jest operacja na liczbie umieszczonej w „st(0)" i podanej jako argument instrukcji. Za każdym razem wynik umieszczany jest w „st(0)". Jeżeli korzystamy z liczby umieszczonej w pamięci to do każdej użytej instrukcji: „fld" (ładowanie do „st(0)"), „fadd" (dodawanie), „fsub" (odejmowanie), „fmul" (mnożenie), „fdiv" (dzielenie) na samym końcu należy dodać sufiks oznaczający typ danych znajdujący się w pamięci – „s" dla pojedynczej precyzji, „l" dla podwójnej, „t" dla podwójnej rozszerzonej.

Częścią zadania było wygenerowanie wyjątków – „NaN", „+/- 0", „+/- inf" (każdy z wyjątków umieszczony jest w „st(0)"):

- NaN – otrzymano wykonując działanie 0/0

```
st0            -nan(0xc000000000000000) (raw 0xffffc000000000000000)
st1            0        (raw 0x00000000000000000000)
st2            0        (raw 0x00000000000000000000)
st3            0        (raw 0x00000000000000000000)
st4            0        (raw 0x00000000000000000000)
st5            0        (raw 0x00000000000000000000)
st6            0        (raw 0x00000000000000000000)
st7            0        (raw 0x00000000000000000000)
fctrl          0x37f    895
fstat          0x3801   14337
ftag           0xbfff   49151
fiseg          0x0      0
fioff          0x600115 6291733
foseg          0x0      0
fooff          0x600088 6291592
fop            0x0      0
mxcsr          0x1f80   [ IM DM ZM OM UM PM ]
```

- +0 – otrzymano wykonując działanie 0/1.5432

```
st0            0        (raw 0x00000000000000000000)
st1            0        (raw 0x00000000000000000000)
st2            0        (raw 0x00000000000000000000)
st3            0        (raw 0x00000000000000000000)
st4            0        (raw 0x00000000000000000000)
st5            0        (raw 0x00000000000000000000)
st6            0        (raw 0x00000000000000000000)
st7            0        (raw 0x00000000000000000000)
fctrl          0x37f    895
fstat          0x3800   14336
ftag           0x7fff   32767
fiseg          0x0      0
fioff          0x600115 6291733
foseg          0x0      0
fooff          0x600088 6291592
fop            0x0      0
mxcsr          0x1f80   [ IM DM ZM OM UM PM ]
```

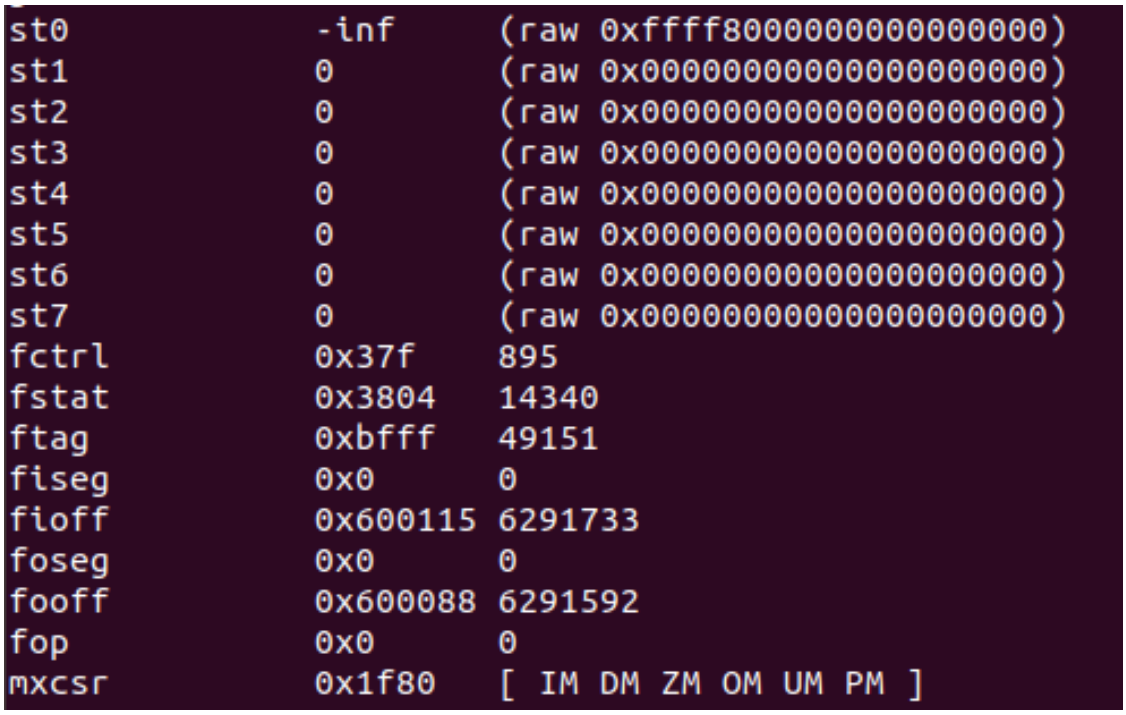
- -0 – otrzymano wykonując działanie 0/-1.5432

```
st0            -0        (raw 0x80000000000000000000)
st1            0         (raw 0x00000000000000000000)
st2            0         (raw 0x00000000000000000000)
st3            0         (raw 0x00000000000000000000)
st4            0         (raw 0x00000000000000000000)
st5            0         (raw 0x00000000000000000000)
st6            0         (raw 0x00000000000000000000)
st7            0         (raw 0x00000000000000000000)
fctrl          0x37f     895
fstat          0x3800    14336
ftag           0x7fff    32767
fiseg          0x0       0
fioff          0x600115  6291733
foseg          0x0       0
fooff          0x600088  6291592
fop            0x0       0
mxcsr          0x1f80    [ IM DM ZM OM UM PM ]
```

- +inf – otrzymano wykonując działanie 1.5432/0

```
st0            inf       (raw 0x7fff8000000000000000)
st1            0         (raw 0x00000000000000000000)
st2            0         (raw 0x00000000000000000000)
st3            0         (raw 0x00000000000000000000)
st4            0         (raw 0x00000000000000000000)
st5            0         (raw 0x00000000000000000000)
st6            0         (raw 0x00000000000000000000)
st7            0         (raw 0x00000000000000000000)
fctrl          0x37f     895
fstat          0x3804    14340
ftag           0xbfff    49151
fiseg          0x0       0
fioff          0x600115  6291733
foseg          0x0       0
fooff          0x600088  6291592
fop            0x0       0
mxcsr          0x1f80    [ IM DM ZM OM UM PM ]
```

- -inf – otrzymano wykonując działanie 1.5432/0



Opis uruchomienia programu

Program został złożony (assembly) i zlinkowany za pomocą pliku „makefile” i komendy „make”. Do debuggowania, kontrolowania zawartości rejestrów i stosu (wyników) posłużono się debuggerem „gdb”. Wyniki działaniu programu można otrzymać następująco: włączyć debugger „gdb”, ustawić breakpoint na etykietę „exit”, rozpocząć przejście przez program i wyświetlić rejestry zmiennoprzecinkowe za pomocą komendy „info all-r”. Zawartość pliku „makefile”:

```
all: kalkulator

kalkulator.o: kalkulator.s
	as -g -o kalkulator.o kalkulator.s

kalkulator: kalkulator.o
	ld -g -o kalkulator kalkulator.o
```