Wrocław, 11 marca 2018

Marcin Kotas, 235098 WT-P-7.15

prowadząca: mgr Aleksandra Postawka

Laboratorium Architektury Komputerów
(1) Utrwalenie umiejętności konstruowania programów wykorzystujących pętle i sprawdzanie warunków.

1 Treść ćwiczenia

Zakres i program ćwiczenia:

Utworzenie programu wczytującego liczbę ze standardowego strumienia i drukującego dolne przybliżenie pierwiastka kwadratowego z tej liczby.

2 Kod programu

2.1 Zapisanie liczby z bufora do rejestru

W pętli to_number wykonuje się bajt po bajcie sprawdzanie, czy znak jest kodem cyfry. Jeśli tak, to odejmowany jest kod znaku ascii '0', po czym cyfra dodawana jest do całej liczby (znajdującej się w rejestrze rax) przemnożonej przez 10. W ten sposób liczba w rejestrze odpowiada liczbie wprowadzonej na klawiaturze.

```
    movq %rax, %r8
    dec %r8             # -'\n'
    movq $0, %rbx
    movq $0, %rdi       # iterator
    movq $10, %r10      # podstawa
    movq $0, %rax       # liczba

    to_number:
        movb textin(, %rdi, 1), %bl
        cmp $'0', %bl
        jl not_number
        cmp $'9', %bl
        jg not_number

        sub $'0', %bl

        mul %r10
        add %rbx, %rax

        inc %rdi
        cmp %r8, %rdi
        jl to_number
```

2.2 Obliczenie pierwiastka

Następnie w pętli square_root obliczana jest suma kolejnych liczb nieparzystych. W momencie, kiedy suma przekroczy wartość wprowadzonej liczby, ilość dodanych liczb nieparzystych wyznacza górne oszacowanie pierwiastka. Aby otrzymać dolne oszacowanie wartość ta jest dekrementowana.
Pętla to_stack wyłuskuje z wyniku kolejne cyfry dzieląc przez 10, po czym dodaje do nich kod znaku '0' oraz wrzuca na stos. Na koniec w pętli to_text przenosi ze stosu cyfry wyniku do bufora textout, aż do uzyskania pełnego wyniku.

```
    movq $0, %r8    # pierwiastek
    movq $1, %rdi
# kolejne l. nieparzyste
    movq $0, %r12   # suma -||-

    square_root:
        add %rdi, %r12
        add $2, %rdi
        inc %r8
        cmp %rax, %r12
        jle square_root

    dec %r8 # dolne oszacowanie

    movq %r8, %rax
    movq $0, %r8
    to_stack:
        div %r10
        add $'0', %rdx
        push %rdx
        movq $0, %rdx
        inc %r8
        cmp $0, %rax
        jg to_stack

    movq $0, %rdi
    to_text:
        pop textout(, %rdi, 1)
        inc %rdi
        cmp %r8, %rdi
        jl to_text

    movb $'\n', textout(, %rdi, 1)
```

3 Wnioski

Program uruchomił się poprawnie. Pętla zamieniająca liczbę na znaki ascii pobiera cyfry od końca, przez co w buforze cyfra zapisana jest od prawej do lewej. Aby zamienić kolejność cyfr wykorzystany został stos.