Wrocław, 25.03.2018

Adam Karczewski, 235039
WT-N-7

prowadzący: Aleksandra Postawka

Laboratorium Architektury Komputerów

(1) Pętle, podstawowe operacje logiczne i arytmetyczne

1.1 Treść ćwiczenia

Zakres i program ćwiczenia:
Nauka sposobu działania podstawowych operacji logicznych i arytmetycznych oraz pętli w języku assembler AT&T

Zrealizowanie zadania:
Pobranie od użytkownika liczby zapisanej w systemie U8, sprawdzenie czy podana liczba jest prawidłowa. Zamiana wartości i wyświetlenie jej w U6.

1.2 Budowa kodu źródłowego

Pierwszym etapem było wczytanie danych ze standardowego strumienia wejścia do buforu textin.

```
movq %rax, %r8
sub $2, %r8
movq $0, %rdi
movq $1, %rsi
```

Rejestr r8 to licznik pętli, kopiujemy do niego długość wczytanego łańcucha. Odejmujemy 2, aby pozbyć się znaku nowej linii z końca łańcucha oraz ze względu na liczenie od 0, a nie od 1. W rejestrze rdi przechowywany jest wynik, a w rsi wytwarzamy kolejne potęgi podstawy wejścia- 8.

```
zamiana:
movq $0, %rax
movb textin(, %r8,1), %al
sub $P_LICZB,%rax
movq %rax, %r9
cmp $0, %rax
jl blad
cmp $7, %rax
jg blad
mul %rsi
```

```
add %rax, %rdi
movq $8, %rax
mul %rsi
movq %rax, %rsi
dec %r8
cmp $0, %r8
jge zamiana
```

Pobieramy znak do rejestru, odejmujemy kod ASCII cyfry 0, od teraz w rejestrze rax jest liczba. Sprawdzamy, czy podane zostały tylko cyfry z zakresu 0-7. W przeciwnym wypadku wyświetlamy komunikat o błędzie. Wymnażamy liczbę z rejestru rax, przez kolejną potęgę 8. Dekrementujemy licznik. Pętlę powtarzamy, dopóki licznik jest >=0.

```
cmp $4, %r9
jl dalej
movq $-1, %rax
mul %rsi
add %rax, %rdi
```

Sprawdzamy znak liczby, jeżeli zaczyna się ona od 4 lub więcej należy dopisać -1 na najwyżej pozycji. W przeciwnym przypadku nie potrzeba wykonywać żadnych operacji. Od tego momentu posiadamy w rejestrze rdi prawidłową liczbę w systemie dziesiętnym.

```
movq %rdi, %rax
movq $6, %rbx
movq $0, %r9
cmp $0, %rax
jge dodatnie
jmp ujemne
```

Następnym etapem jest sprawdzenie znaku liczby- od tego zależy sposób konwersji na system U6. Dodatkowo szykujemy licznik- r9, który będzie przydatny podczas zamiany oraz kopiuję podstawę systemu- 6 do rejestru rbx.

```
dodatnie:
movq $0, %rdx
div  %rbx
add $P_LICZB, %dl
movb %dl,text(,%r9,1)
inc %r9
cmp $0,%rax
jne dodatnie
```

Jeżeli liczba była dodatnia wykonuję operację mod 6. Wynik koduję w ASCII i zapisuję do bufora text. Operację powtarzam dopóki wynik dzielenia przez 6 liczby zapisanej w rejestrach rdx:rax nie będzie 0. Podczas dzielenia wynik zapisywany jest do rax, a reszta do rdx.

```
ujemne:
movq $-1, %rdx
idiv %rbx
add $6,%rdx
add $P_LICZB, %rdx
movb %dl, text(,%r9,1)
```

```
inc %r9
cmp $0,%rax
je enter

ujemne_petla:
movq $-1, %rdx
idiv %rbx
add $5,%rdx
add $P_LICZB, %rdx
movb %dl, text(,%r9,1)
inc %r9
cmp $0, %rax
jne ujemne_petla
```

Aby otrzymać prawidłowy wynik konwersji dla liczby ujemnej należy wykonać kilka dodatkowych operacji. Dzielenie ze znakiem opisuje mnemonik idiv, aby wynik był poprawny należy dopisać rozszerzenie tj. wstawić -1 do rejestru rdx. W ten sposób otrzymujemy wynik zapisany w systemie znak-moduł. Aby dokonać konwersji na system uzupełnieniowy należy otrzymany wynik odjąć od -1. Co w praktyce oznacza zamianę wyniku przy najmniejszej potędze (6^0) po przez dodanie 6. Na następnych pozycjach dodajemy 5. Tak jak poprzednio kodujemy liczbę w ASCII.

```
odwroc_dodatnie:
movq $0, %r10
movb $'0',textout(,%r10,1)
movq $1, %rdi
dec %r9
```

Kolejnym krokiem jest dopisanie rozszerzenia liczby (odpowiednio 0 lub 5). Wynik w buforze text jest zapisany w odwrotnej kolejności. Odwracamy go- korzystając z liczników rdi- licznik od początku i r9- licznik od końca.

```
odwroc_petla:
movb text(,%r9,1), %al
movb %al, textout(,%rdi,1)
dec %r9
inc %rdi
cmp $0, %r9
jge odwroc_petla
jmp enter
```

W pętli kolejno pobieramy dane od końca bufora text i zapisujemy je do bufora textout. Ostatnim krokiem jest wyświetlenie wyniku i zakończenie wykonywania programu.

1.3 Wnioski

W trakcie trwania laboratorium wykonano część programu- pobranie i sprawdzenie poprawności wprowadzonych danych. Resztę zadania dokończono w domu. Program przetestowano- działa prawidłowo.