Wrocław, 23.04.2018

Adam Karczewski, 235039
WT-N-7

prowadzący: Aleksandra Postawka

Laboratorium Architektury Komputerów

(3) Funkcje

3.1 Treść ćwiczenia

Zakres i program ćwiczenia:
Celem ćwiczenia była nauka tworzenia funkcji w języku assemblera. Na laboratorium zapoznaliśmy się z przekazywaniem i zwracaniem argumentów funkcji zarówno przez stos jak i przez rejestry.

Zrealizowanie zadania:
Pierwszym zadaniem było napisanie funkcji, która zwraca adres początku najdłuższego ciągu zer w podanym jako argument łańcuchu znakowym. Drugie zadanie polegało na napisaniu funkcji, która oblicza rekurencyjnie wartość n-tego wyrazu ciągu:

$$x_n = \begin{cases} x_0 = -2 \\ x_n = 5x_{n-1} + 7 \end{cases}$$

Zadana funkcja miała przyjmować i zwracać argumenty na dwa sposoby: przez stos i za pomocą rejestrów.

3.2 Budowa kodu źródłowego

3.2.1 Adres najdłuższego ciągu zer

Funkcja przyjmuje jako argumenty adres początku łańcucha oraz jego długość. Argumenty przekazane są przez stos. Funkcja zostaje wywołana w poniższy sposób.

```
push $znak_len
push $znak
call f
```

Argumenty umieszczamy na stosie instrukcją push, następnie wywołujemy funkcję f instrukcją call. Na początku funkcji należy zabezpieczyć adres powrotu z funkcji, który znajduje się na szczycie stosu. Wskaźnik bieżącej ramki stosu umieszczamy na jego szczycie, a następnie zmieniamy jego wartość kopiując zawartość z rejestru rsp- wskaźnik szczytu stosu.

```
push %rbp
movq %rsp, %rbp
```

Kolejnym krokiem jest ściągnięcie ze stosu argumentów funkcji- należy pamiętać, że stos jest strukturą typu FIFO, ściągamy argumenty w odwrotnej kolejności niż je wkładaliśmy.

```
movq 16(%rbp), %rax
movq 24(%rbp), %r10
```

Stos wygląda następująco:

$znak_len
$znak
adres powrotu
„stary” wskaźnik stosu

Ponieważ każdy element na stosie jest 8 bajtowy, dlatego adres pierwszego argumentu funkcji wynosi 16*rbp. W rejestrze rax zostaje zapisany początek ciągu badanych znaków, a w r10 jego długość.

```
movq $0, %rbx
movq $0, %rdx
movq $-1, %rdi
movq $0, %r8
movq $-1, %r9
```

Przygotowujemy rejestry do badania zadanego łańcucha. Rejestr rbx jest licznikiem pętli, rdx licznikiem ilości zer w podciągu, rdi- licznikiem pozycji w podciągu zer, r8 zapamiętuje maksimum wystąpień zer, r9 początek najdłuższego podciągu.

```
licz:
movb (%rax,%rbx,1), %cl
cmp $'0', %cl
jne k_zer
inc %rdx
movq %rbx, %rdi
jmp k_petli
```

Pobieramy znak z łańcucha- korzystając z trybu adresowania. Sprawdzamy, czy znak jest równy ‘0’, jeżeli tak zwiększamy rejestr wystąpień zer- rdx i zapisujemy adres zera w rejestrze rdi.

```
k_zer:
cmp %r8, %rdx
jle k_petli
movq %rdx, %r8
```

```
dec %rdx
sub %rdx, %rdi
movq %rdi, %r9
movq $0, %rdx
movq $-1, %rdi
```

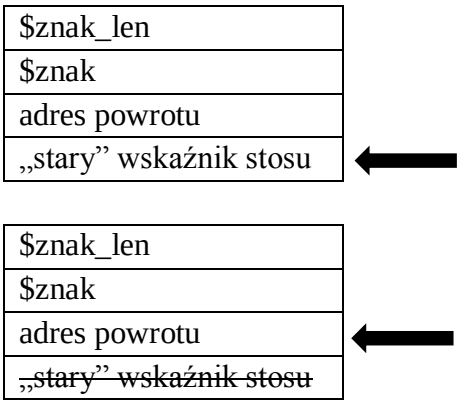
Jeżeli pobrany znak nie jest zerem sprawdzamy, czy nie jest to pierwszy znak występujący po najdłuższym podciągu zer. Długość dotychczasowego maksymalnego ciągu zapisane jest w rejestrze r8, a ilość zer bezpośrednio poprzedzających znak jest zapisana w rdx. Jeżeli znaleźliśmy nowe maksimum zapisujemy jego długość do r8 oraz obliczamy jego początek odejmując od bieżącej pozycji ilość zer. Wynik zapisujemy w r9 oraz zerujemy liczniki rdx i rdi.

```
k_petli:
inc %rbx
cmp %r10, %rbx
jle licz
```

Każdy obrót pętli kończy się inkrementacją licznika i sprawdzeniem warunku zakończenia pętli.

```
movq %r9, %rax
movq %rbp, %rsp
pop %rbp
ret
```

Wynik zwracamy przez rejestr rax. Pobieramy ze stosu adres powrotu i kończymy funkcję instrukcją ret.



3.2.2 Funkcja rekurencyjna- przekazanie parametrów przez rejestry

Przed wywołaniem funkcji umieszczamy numer wyrazu w rejestrze rax. Na początku funkcji zabezpieczmy adres powrotu w analogiczny jak poprzednio sposób.

```
cmp $0, %rax
jne rekurencja
movq $-2, %r8
jmp koniec_f
```

Sprawdzamy, czy numer wyrazu jest równy 0, jeśli tak to do rejestru r8 zapisujemy wartość pierwszego elementu ciągu: -2. Jeżeli obliczany wyraz ma numer większy niż 0 wywołujemy funkcję rekurencyjnie.

```
dec %rax
call funkcja
movq %r8, %rax
movq $5, %r8
mul %r8
movq %rax, %r8
add $7, %r8
```

Funkcja wywołuje samą siebie z parametrem n-1. Zwrócony wynik w rejestrze r8 zgodnie ze wzorem mnożymy razy 5 i dodajemy 7.

Funkcja kończy się przywróceniem stanu stosu jak w poprzednim przypadku. Wynik działania funkcji jest w rejestrze r8.

3.2.3 Funkcja rekurencyjna- przekazanie parametrów przez stos

Przekazanie parametrów odbywa się tak jak w pierwszym zadaniu. Reszta funkcji jest analogiczna do wersji z przekazywaniem przez rejestr. Różni się tylko zwrócenie wyniku.

```
movq %r8, 16(%rsp)
```

Wynik znajdujący się w rejestrze r8 umieszczamy tak, aby po zakończaniu funkcji był na jego szczycie i można go było pobrać instrukcją pop.

wynik
adres powrotu
„stary” wskaźnik stosu

3.3 Wnioski

Podczas zajęć nauczyłem się tworzyć funkcję w języku assemblera oraz poznałem różne sposoby przekazywania argumentów i wyników.