Примеры программ для модельного ассемблера

Коновалов А.В.

18 февраля 2023 г.

Вычисление факториала

Программа, которую мы хотим написать, должна считывать со стандартного ввода число и выводить факториал этого числа на стандартный вывод. Ситуация усложняется тем, что нам придётся написать код для ввода и вывода целых чисел, т.к. ввод-вывод у нас поддерживается только посимвольный.

В комментариях будем записывать содержимое стека. Адрес возврата будем обозначать как `ret`

```
main CALL
HALT

:main          ; ret
read_int CALL  ; ret x
factorial CALL ; ret x!
write_int CALL ; ret
0 SWAP         ; 0 ret
GOTO           ; 0

:read_int      ; ret
0              ; ret 0

:read_int_loop ; ret x
IN DUP         ; ret x c c
48             ; ret x c c 48
CMP            ; ret x c sgn
read_int_exit JLT
DUP 57 CMP     ; ret x c sgn
read_int_exit JGT
48 SUB         ; ret x d=(c-'0')
SWAP           ; ret d x
10 MUL         ; ret d x*10
```

```
ADD            ; ret x*10+d
read_int_loop GOTO

:read_int_exit ; ret x c
DROP SWAP      ; x ret
GOTO           ; x

:factorial     ; n ret
SWAP           ; ret n
DUP factorial_non_zero JNE
DROP 1 SWAP    ; 1 ret
GOTO           ; 1

:factorial_non_zero
DUP 1 SUB      ; ret n (n-1)
factorial CALL ; ret n (n-1)!
MUL SWAP       ; n*(n-1)! ret
GOTO           ; n*(n-1)!

:write_int     ; x ret
SWAP           ; ret x
DUP 10 MOD     ; ret x x%10
SWAP 10 DIV    ; ret x%10 x/10
DUP            ; ret x%10 x/10 x/10
write_int_skip_prefix JEQ
               ; ret x%10 x/10
write_int CALL ; ret x%10
100500         ; ret x%10 100500
:write_int_skip_prefix
               ; ret x%10 ?
DROP           ; ret x%10
48 ADD OUT     ; ret
GOTO           ; пусто
```

Другой пример. Напишем программу, которая считывает со стандартного ввода несколько чисел и вычисляет их сумму.

Формат входных данных:

`N x0 ... xN-1`

В начале записывается количество чисел, затем сами числа. Числа разделяются одним знаком пробела.

Для наглядности воспользуемся массивом-локальной переменной.

```
main CALL
HALT
```

```
:main           ; ret
GETSP SETBP     ; ret
                ; регистр BP теперь указывает на адрес возврата
read_int CALL   ; ret N
DUP             ; ret N N
PUSHN           ; ret N x0 ... xN-1
0               ; ret N x0 ... xN-1 0

; Регистр BP указывает на слово, где лежит адрес возврата ret.
; Стек растёт в сторону младших адресов. Адрес слова, где лежит
; N --- BP-1, где лежит x0 --- BP-2, xi --- BP-2-i,
; BP-1-N, счётчик цикла i --- BP-2-N. Но счётчик лежит на вершине
; стека, поэтому к нему мы можем обращаться непосредственно.

:main_read_loop ; ret N x0 ... xN-1 i
DUP             ; ret N x0 ... xN-1 i i
GETBP 1 SUB READ ; ret N x0 ... xN-1 i i N
CMP main_read_loop_exit JEQ
                ; ret N x0 ... xN-1 i
IN DROP         ; пропускаем пробел
read_int CALL   ; ret N x0 ... xN-1 i xi
OVER            ; ret N x0 ... xN-1 i xi i
GETBP 2 SUB     ; ret N x0 ... xN-1 i xi i &x0
SWAP SUB        ; ret N x0 ... xN-1 i xi &xi
WRITE           ; ret N x0 ... xN-1 i
1 ADD           ; ret N x0 ... xN-1 i+1
main_read_loop GOTO

:main_read_loop_exit ; ret N x0 ... xN-1 N=i
DROP                 ; ret N x0 ... xN-1
0 0                  ; ret N x0 ... xN-1 0 0

:main_sum_loop       ; ret N x0 ... xN-1 sum i
DUP                  ; ret N x0 ... xN-1 sum i i
GETBP 1 SUB READ     ; ret N x0 ... xN-1 sum i i N
CMP main_sum_loop_exit JEQ
                     ; ret N x0 ... xN-1 sum i
GETBP 2 SUB          ; ret N x0 ... xN-1 sum i &x0
OVER SUB             ; ret N x0 ... xN-1 sum i &xi
READ                 ; ret N x0 ... xN-1 sum i xi
ROT                  ; ret N x0 ... xN-1 i xi sum
ADD                  ; ret N x0 ... xN-1 i sum+xi
SWAP 1 ADD           ; ret N x0 ... xN-1 sum+xi i+1
main_sum_loop GOTO

:main_sum_loop_exit  ; ret N x0 ... xN-1 sum N=i
```

```
GETBP 1 SUB          ; ret N x0 ... xN-1 sum N=i &N
                     ;     ^----------------------/
ROT                  ; ret N x0 ... xN-1 N &N sum
                     ;     ^---------------/
WRITE                ; ret sum x0 ... xN-1 N
DROPN                ; ret sum
write_int CALL       ; ret
0 SWAP GOTO          ; 0

:read_int
... см. выше ...

:write_int
... см. выше ...
```