PROGRAMOWANIE NA POZIOMIE ASEMBLERA

ZADANIA

Z1. Sprawdź działanie środowiska. Wklej poniższy program do pliku *hello.asm*. Przeanalizuj plik, skompiluj i uruchom (./compile-run-hello.sh).

```
%include "AuxMacros.asm"
section     .text
global _start
_start:
      mov dx,258      ; dx = 258;
      printReg dx     ; printf("%h\n", dx);
      return0         ; return 0;
section     .data
HexDig      db '0', '1', '2', '3'
      db '4', '5', '6', '7'
      db '8', '9', 'A', 'B'
      db 'C', 'D', 'E', 'F'
msg   db '12 = 0000', 0xa
len     equ $ - msg
```

Z2. Uruchom poniższy program. Zmodyfikuj dane wejściowe (*ax*), sprawdź jak zadziała program dla ujemnych danych wejściowych (-1, -5 …).

```
%include "AuxMacros.asm"
section     .text
global _start
_start:
                  ; ax = 3 * ax;
      mov ax, 2   ; input data
      mov bx, ax
      add ax, ax
      add ax, bx
      printReg ax ; printf(("%h\n", ax);
      return0     ; return 0;
section     .data
HexDig      db '0', '1', '2', '3'
      db '4', '5', '6', '7'
      db '8', '9', 'A', 'B'
      db 'C', 'D', 'E', 'F'
msg   db '12 = 0000', 0xa
len     equ $ - msg
```

Z3. Uruchom poniższy program. Zmodyfikuj argumenty, wypróbuj różne kombinacje ujemnych i dodatnich wartości ({3,-5}, {-3, 5}, {-3,-5} …), przeanalizuj wyniki.

```
%include "AuxMacros.asm"
section     .text
global _start
_start:
                ; ax = min(bx,cx)
      mov bx,4   ; first argument
      mov cx,5   ; second argument
      mov ax, bx
      cmp ax, cx
      jle ok
      mov ax, cx
ok:
      printReg ax    ; printf("%h\n", ax);
      return0        ; return 0;
section     .data
HexDig      db '0', '1', '2', '3'
      db '4', '5', '6', '7'
      db '8', '9', 'A', 'B'
      db 'C', 'D', 'E', 'F'
msg   db '12 = 0000', 0xa
len     equ $ - msg
```

Z4. Uruchom poniższy program. Zmodyfikuj argumenty i przeanalizuj rezultaty. Co się stanie, jeśli co najmniej jeden z argumentów będzie równy 0? Dodaj testy (czy jest 0 w rejestrze i skok) zabezpieczające przed tym problemem.

```
%include "AuxMacros.asm"
section     .text
global _start
_start:
      mov ax, 6   ; first argument
      mov bx, 15  ; second argument

whi: cmp ax, bx
     je kon
     jle els
     sub ax, bx
     jmp od
els: sub bx, ax
od:  jmp whi
kon:
     printReg ax ; printf(("%h\n", ax);
     return0     ; return 0;
section     .data
HexDig      db '0', '1', '2', '3'
      db '4', '5', '6', '7'
      db '8', '9', 'A', 'B'
      db 'C', 'D', 'E', 'F'
msg   db '12 = 0000', 0xa
len     equ $ - msg
```

Z5. Przeanalizuj program *digits-total.asm* (w archiwum *nasm.zip*), uruchom program (./compile-run-digits-total.sh). Zmodyfikuj kod tak, aby użyć innej podstawy systemu i przetestuj swoje rozwiązanie.