Pomnilniško naslavljanje in spremenljivke

Sistemi 1 - teden 3

Vrste naslavljanj:

Vrsta	Primer
Takojšnje naslavljanje	`MOV A, 100`
Registrsko naslavljanje	`MOV A, B`
Neposredno naslavljanje	`MOV A, [100]`
Posredno naslavljanje	`MOV A, [B+100]`

Oglati oklepaji `[]` vedno pomenijo dostop do pomnilnika. Vsak ukaz lahko izvede največ en dostop do pomnilnika.

Naloga 1. Kakšna je razlika med spodnjima programoma?

```
MOV D, 100
MOV A, D
```

in

```
MOV D, 100
MOV A, [D]
```

Rešitev: Prvi program v register A naloži vrednost 100, drugi program pa vrednost, ki se nahaja na pomnilniški lokaciji z naslovom 100 (`0x64`).

Naloga 2. Na pomnilniško lokacijo z naslovom `0x100` shranite šestnajstiško vrednost `abcd`. Kateri pomnilniški naslovi hranijo kateri del te vrednosti? Sedaj na pomnilniško lokacijo `0x102` shranite 8-bitno vrednost `0x33`. Napišite program, ki vrednosti na naslovu `0x100` prišteje vrednost na naslovu `0x102`.

Rešitev:

```
MOV [0x100], 0xabcd ; 16-bit value x
MOVB [0x102], 0x33  ; 8-bit value y

; Implementation of x = x + y
MOV A, [0x100]    ; Get the value of variable x.
MOV B, 0          ; We use B to convert y to a 16-bit value.
MOVB BL, [0x102]  ; Move y to the lower byte of B.
ADD A, B          ; x + y -> A as 16-bit addition.
MOV [0x100], A    ; Store the result x + y to x.
HLT
```

Naslov `0x100` hrani vrednost `ab`, naslov `0x101` pa vrednost `cd`.

Naloga 3. Definirajte 16-bitno spremenljivko x z začetno vrednostjo $0xabcd$, nato pa njeno vrednost povečajte za 3. Spremenljivko x definirajte na sledeče načine:

1. kot konstanten pomnilniški naslov `0x100`,
2. kot oznako oz. *labelo* (angl. *label*) pod programsko kodo,
3. kot labelo nad programsko kodo,
4. kot labelo na pomnilniškem naslovu `0x100`.

Rešitve:

1. Kot konstanten pomnilniški naslov `0x100`.

```
; The programmer decides that value x is stored
; at the memory address 0x100.
MOV [0x100], 0xabcd ; Set x = 0xabcd
; The following three lines implement x = x + 3.
MOV A, [0x100] ; Get the value of x.
ADD A, 3       ; Add 3 to the value.
MOV [0x100], A ; Store the new value back to x.
HLT
```

2. Kot labelo pod programsko kodo.

```
; The programmer does not care about the actual address of variable x,
; let the assembler determine it.
MOV A, [x] ; Get the value from address x.
ADD A, 3   ; Add 10 to the value.
MOV [x], A ; Store the new value back to address x.
HLT        ; Must (!) stop before data begins.
x: DW 0xabcd; Let label x determine the memory address after HLT.
```

3. Kot labelo nad programsko kodo.

```
; Let variable x reside before the program code.
JMP main
x: DW 0xabcd  ; define and initialize x = 1

; let label main determine the address of the MOV opcode below.
main:
    MOV A, [x] ; Get the value from address x.
    ADD A, 3   ; Add 10 to the value.
    MOV [x], A ; Store the new value back to address x.
    HLT

; Let variable x reside at a specific address in memory.
MOV A, [x]
ADD A, 3
MOV [x], A

; Instruction to the compiler where in memory to compile the code below.
ORG 0x100
x: DW 0xabcd
```

Naloga 4. Denimo, da 16-bitna spremenljivka `x` hrani celo predznačeno število, 8-bitna spremenljivka `pos` pa hrani vrednost `true` (1) ali `false` (0). Napišite program, ki spremenljivko `pos` nastavi na `true`, če je `x` pozitivno število ali ničla, ter `false` sicer.

Rešitev:

```
JMP main

; Variables
x: DW 0x0123   ; int x = 0x0123
pos: DB 0      ; char pos = 0

; Main function (program entry point)
main:
    MOV A, [x]     ; Get the value of variable x.
    SHR A, 15      ; Exclude all bits except the sign bit.
    NOT A          ; Negate the shifted sign bit (and all other bits as well).
    AND A, 1       ; Exclude all other bits.
    MOVB [pos], AL ; Store the result in variable pos.
    HLT
```

Zgled za $x < 0$:

A: 1... ....
`SHR A, 15`
A: 0000 0000 0000 0001
`NOT A`
A: 1111 1111 1111 1110
`AND A, 1`
A: 0000 0000 0000 0000

Maskiranje bitov:
Maska določi, katere bite obdržimo (1) in katere izničimo (0).

x	1010 1010 1010 1010
maska	1111 0110 0000 1001
AND	1010 0010 0000 1000

Domača naloga:

Definirajte 16-bitne spremenljivke x, y, in z. Napišite program, ki izračuna $z = z - (x + y)$. Razložite, kako se po zagonu vašega programa vidi, da program deluje pravilno.