Operativni Sistemi

Direktan rad sa IO uređajima

Pomeranje bita

- `sar`: shift arithmetic right
- `sal`: shift arithmetic left
- `shr`: (bitwise) shift right
- `shl`: (bitwise) shift left

- Kod aritmetičkog pomeranja desno se zadržava znak.

  >>> shifts.asm

Komunikacija sa I/O uređajem

- Ispis na proizvoljnom I/O uređaju se vrši pomoću instrukcije `out`, npr:

```
out dx, al
```

- Gde `dx` označava adresu uređaja, a `al` proizvoljan argument.
- Kada se adresa uređaja navede kao konstanta, argument mora da bude jedan bajt. Ako se adresa navodi kroz registar (kao gore `dx`), parametar može da bude jedan bajt (npr `al`), dva bajta (npr `ax`) ili četiri bajta (`eax`)
- Analongno funkcioniše instrukcija `in`:

```
in al, dx
```

Komunikacija sa štampačem

- Štampač se nalazi na paralelnom portu, adresa `378h`.
- Parametar koji se navodi je slovo koje želimo da se ispiše
- Na narednoj adresi može da se pročita ready flag na najznačajnijem bitu (`X-------`), a na adresi posle nje se nalazi Strobe (lampica) na najmanje značajnom bitu (`-------X`)

>>> printer.asm (prva polovina)

Interrupt flag

- Svaki interrupt ima svoju oznaku (broj) i svoj hendler (kod koji se izvrši po pozivu interrupt-a). Hendleri mogu da se izmene u toku rada računara.
- Interrupt flag služi da omogući, tj. onemogući rad prekidnih rutina. Kada se interrupt flag postavi na nulu (clear), prekidi nisu mogući, sve dok se flag ne postavi opet na jedinicu (set). Onemogućavanje prekida se koristi kada treba postaviti novi hendler.
- `cli` – clear interrupt
- `sti` – set interrupt

Tabela prekida

- U nultom segmentu se nalazi tabela prekida. U njoj je zapisana adresa koda koji treba izvršiti za svaki pojedinačni prekid. Svaki prekid je opisan sa četiri bajta (segment i offset). Dakle, ako želimo da izmenimo hendler za prekid `10`, treba da na adresu `[0:10*4+2]` upišemo segment hendlera, i na adresu `[0:10*4]` upišemo ofset unutar segmenta na kom počinje kod.
- Za naznačavanje offseta, možemo da koristimo labele.
- Prekidna rutina se završava komandom iret

  >>> printer.asm (druga polovina)

Tastatura – scan code

- Pri pritisku i puštanju tastera, tastatura šalje scan code na osnovu kog se identifikuje koji taster je pritisnut tj. otpušten.
- Scan code za pritisak tastera nije isti kao scan code za otpuštanje tog tastera.

>>> scan.asm

Scan code to ASCII

- Ako hoćemo da vidimo koji znak je pritisnut, moramo da izvršimo konverziju u ASCII.
- Ovo ne može da se izvede „lepo“ kao npr konverzija malih ASCII slova u velika.
- Koristi se translaciona tabela.

Translaciona tabela

- Ideja je da se na nekoj lokaciji u memoriji upišu redom vrednosti ASCII kodova za odgovarajuće sken kodove (počevši od nultog).
- Instrukcija `xlat` vrši konverziju bajta koji se nalazi u `al` prema translacionoj tabeli koja se nalazi na adresi `DS:BX`

>>> kbd.asm

Vežba

- Napraviti osnovu za igru “zmija”:
  - Zmija fiksne dužine se kreće u određenom pravcu na ekranu.
  - Kada korisnik pritisne strelicu gore, dole, levo ili desno na tastaturi, zmija skrene u odgovarajućem pravcu i nastavlja da se kreće u tom pravcu.
  - Obezbediti da se zmija pojavljuje na suprotnoj strani ekrana ako izađe izvan okvira.