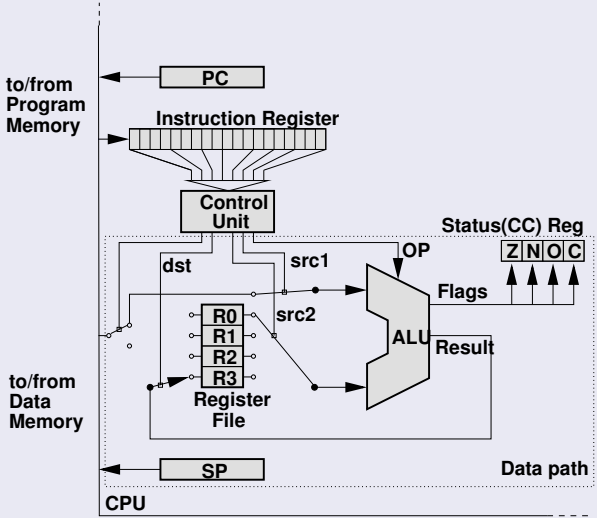
Mikroprocesorski sistemi

Arhitektura mikrokontrolera

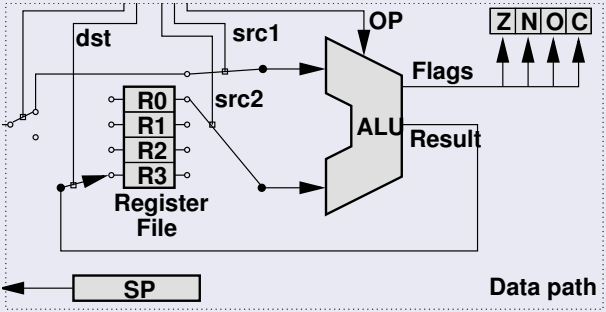
dr Srđan T. Mitrović, dipl.inž.

e-mail: srdjan.mitrovic@va.mod.gov.rs
telefon: 40–681, kancelarija 131

Osnovna arhitektura procesora



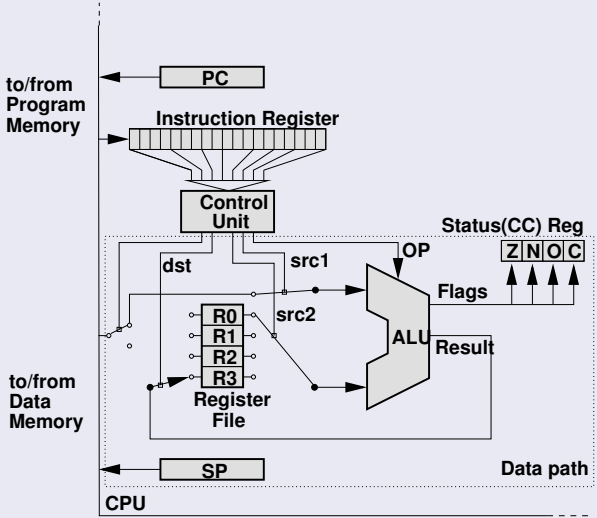
Data path



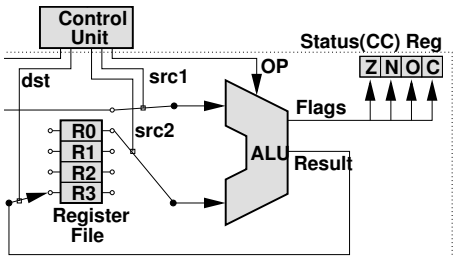
Osnovu arhitekture čine:

- *Data path*
- *Control Unit*

Osnovna arhitektura procesora



Aritmetičko logička jedinica – ALU



Namena ALU:

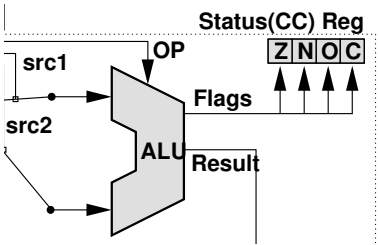
- izvršenje operacija (AND, ADD, INC, . . .)

Uzima dva ulaza i kao rezultat vraća jedan izlaz

Ulazni podaci se uzimaju iz registara ili iz memorije

Izlaz se smešta u registar ili u memoriju

Statusni registar



Osim rezultata operacije, generišu se podaci o njegovoj prirodi:

- **Z (Zero)**: Rezultat operacije je nula
- **N (Negative)**: Rezultat operacije je negativan
- **O (Overflow)**: Operacija je generisala prekoračenje
- **C (Carry)**: U operaciji postoji prenos

Predstavljanje negativnih brojeva

Računar za predstavljanje brojeva koristi 0 i 1. Kako predstaviti negativan ceo broj?

Ideja 1: Invertovati sve bitove pozitivnog broja. Ovo se zove komplement jedinice.

Nedostatak: Postoje dve interpretacije nule.

Ideja 2: Invertovati sve bitove pozitivnog broja i dodati 1. Ovo se zove komplement dvojke.

Primer

Za četvorobitnu reprezentaciju se dobija:

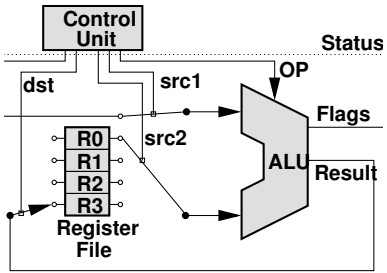
$$1 = 0001 \rightarrow -1 = 1110 + 1 = 1111 \tag{1}$$

$$0 = 0000 \rightarrow -0 = 1111 + 1 = 0000 \tag{2}$$

Registar File

Sadrži radne registre CPU. Registri mogu biti opšte namene (*general purpose*), u kojima se smeštaju operandi ili rezultat operacije. Postoje i specijalni registri (*dedicated register*):

- Akumulator(*accumulator*): koristi se za aritmetičke i logičke operacije
- (*Index register*): koristi se za neke modove adresiranja
- ...



Upotreba statusnog registra

Statusni regustar se koristi pri sabiranju ili oduzimanju brojeva čija dužina je veća od procesorske reči.
CPU nudi instrukcije koje mogu da koriste, na primer, (*carry flag*), kao što je **ADDC** – saberi sa prenosom.

Primer: `0x01F0+0x0220`

```
CLC             ; briše carry flag
LD R0,   #0xF0 ; učitava prvi niži bajt u R0
ADDC R0, #0x20 ; sabira drugi niži bajt sa
                ; prenosom (carry <- 1)
LD R1,   #0x01 ; učitava prvi viši bajt u R1
ADDC R1, #0x02 ; sabira drugi viši bajt, prenos
                ; iz prethodnog ADDC je dodat
```

Stek (*stack*) je deo memorije u prostoru podataka, koji koristi CPU za smeštaj adrese povratka i eventualno sadržaja registara u slučaju:

- poziva procedure
- dešavanja prekida

Komande za pristup steku su:

- `PUSH` (stavi nešto u stek) i
- `POP` (uzmi nešto sa steka).

Stack pointer (`SP`) – pokazivač steka

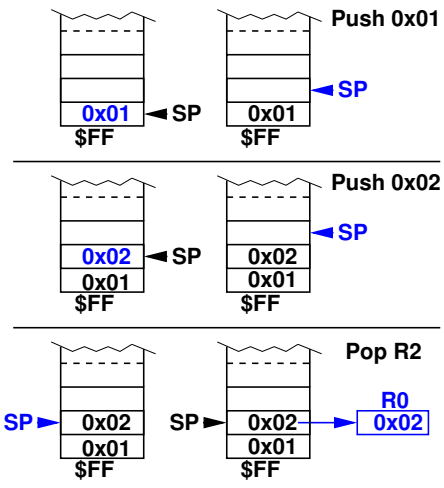
Na žalost, `SP` može na dva načina pokazivati na memorijsku lokaciju u steku:

- prva slobodna adresa (Atmel AVR)
- poslednja korišćena adresa

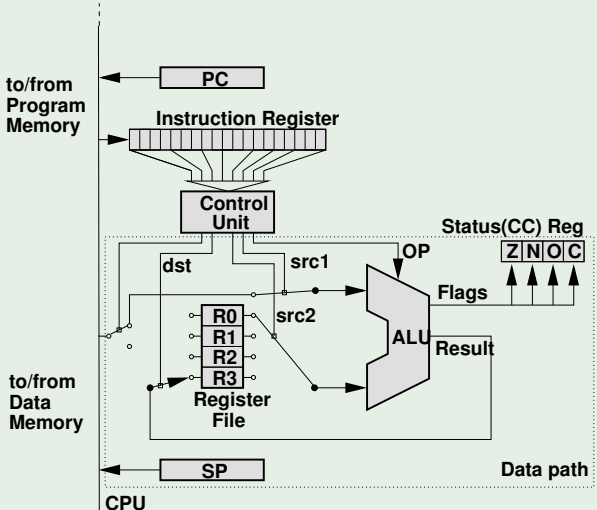
Napomena

`SP` inicijalizuje programer, morate se upoznati sa načinom na koji ga Vaš kontroler koristi.

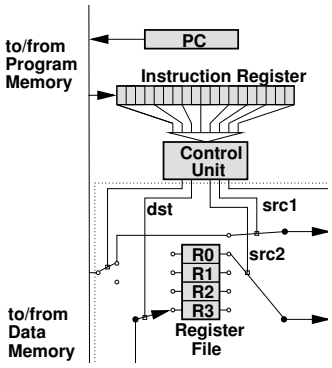
Primer


Push 0x01
0x01
SP
$FF
0x01
SP
$FF
Push 0x02
0x02
0x01
SP
$FF
0x02
0x01
SP
$FF
Pop R2
SP
0x02
0x01
$FF
SP
0x02
0x01
$FF
R0
0x02

Upravljačka jedinica (Control Unit)



Upravljačka jedinica (Control Unit)



- CPU neprekidno izvršava instrukcije (sem pri resetu i HALT).
- Upravljačka jedinica odlučuje koja se instrukcija izvršava,
- Određuje potreban tok podataka.

PC–Program Counter

Čuva adresu sledeće naredbe programa

Instruction Register–IR

Upravljačka jedinica je učitava u registar intrukcija.

Poređenje arhitektura

CISC

Complex Instruction Set Computer

- složene mikroinstrukcije
- veliki broj naredbi
- duže vreme izvršavanja instrukcije

RISC

Reduced Instruction Set Computer

- jednostavna arhitektura
- redukovan broj naredbi
- kraće vreme izvršanja instrukcije

sadržaj registra $D1$ u memoriju na adresu: $24 + [A0] + 4 * [D0]$

CISC: `MOVE D1,([24,A0,4*D0])`

RISC:

```
LD   R1, X   ; indirektno učitava podatak (iz [X] u R1)
LSL  R1      ; pomeranje ulevo -> množenje sa 2
LSL  R1      ; urađeno je 4*[D0]
MOV  X,  R0  ; postavljanje pokazivača (učitavanje A0)
LD   R0, X   ; indirektno učitavanje (završeno [A0] )
ADD  R0, R1  ; sabiranje pokazivača ([A0]+4*[D0])
LDI  R1, $24 ; učitavanje konstante ($ = hex)
ADD  R0, R1  ; i sabiranje (24+[A0]+4*[D0])
MOV  X,  R0  ; postavljanje pokazivača za upis
ST   X,  R2  ; upisivanje vrednosti ([24+[A0]+4*[D0]] <- R2)
```

Instrukcija procesora 68030(CISC) se izvršava u 14 ciklkusa takta, dok se kôd za RISC procesor (Atmel) izvršava u 13 ciklusa takta.

Outline