

Arhitektura – komentirano

Ovaj dokument detaljno opisuje R jezika osnovne značajke, organizaciju podataka, skup instrukcija, kontrolu toka i primjenu u programiranju. Ključni dijelovi za učenje su:

Arhitekture

Reduced instruction set computing (RISC) je jednostavanski skup instrukcija, bržu izvedbu, ali više instrukcija za istu funkcionalnost u usporedbi s CISC arhitekturama.

Arhitektura: operacije se izvode samo na registrima, a podaci iz memorije prvo se učitavaju u registre, obrađuju i zatim pohranjuju natrag.

Memorija organizirani su u bite riječi 2 (4 bajta), a little endian ili big endian načinom.

Arhitektura u R

14 općih namjene, koriste se za aritmetičke i logičke operacije.

Program Counter (PC) sadrži adresu sljedeće instrukcije.

Program Status Register (PSR) :

- (negative bit) – postavlja se ako je rezultat negativan.
- Z (zero bit) – postavlja se ako je rezultat 0.
- (carry bit) – postavlja se pri preljevu rezultata.
- V (overflow bit) – postavlja se ako dolazi do prekoračenja u izračunu.

Arhitekture RISC

Aritmetičke operacije

- ADD $r0, r1, r2 \rightarrow r0 = r1 + r2$
- SUB $r0, r1, r2 \rightarrow r0 = r1 - r2$
- MUL $r0, r1, r2 \rightarrow r0 = r1 * r2$
- RSB $r0, r1, r2 \rightarrow r0 = r2 - r1$ (oduzimanje u obrnutom redoslijedu)

logičke operacije

- `AND r0, r1, r2` → $r0 = r1 \text{ AND } r2$
- `ORR r0, r1, r2` → $r0 = r1 \text{ OR } r2$
- `EOR r0, r1, r2` → $r0 = r1 \text{ XOR } r2$
- `BIC r0, r1, r2` → $r0 = r1 \text{ AND (NOT } r2)$ (bit-clear operacija)

omicanje bitova

- `LSL r0, r1, #2` → pomakni sadržaj r1 ulijevo za 2 bita (ekvivalentno množenju s 2^2).
- `LSR r0, r1, #2` → pomakni sadržaj r1 udesno za 2 bita (ekvivalentno dijeljenju s 2^2).
- `ASR r0, r1, #2` → aritmetički pomak udesno (zadržava predznak).

sporedbe

- `CMP r0, r1` → usporedi r0 i r1, postavlja CPSR statusne bitove (ali ne mijenja registre).

premještanje podataka

- `MOV r0, r1` → kopira sadržaj r1 u r0.
- `MVN r0, r1` → kopira negiranu vrijednost r1 u r0.

čitavanje podataka iz memorije i registara

- `LDR r0, [r1]` → učitava sadržaj memorijske lokacije iz r1 u r0.
- `STR r0, [r1]` → pohranjuje sadržaj r0 u memorijsku lokaciju pohranjenu u r1.

adresiranje

- `LDR r0, [r1, #4]` → $r0 = \text{sadržaj memorijske lokacije na adresi } (r1 + 4)$.
- `LDR r0, [r1, -r2]` → $r0 = \text{sadržaj memorijske lokacije } (r1 - r2)$.

lokalizacija izvora podataka

grananja

- `B label` → bezuvjetno grananje na adresu label.

grananja u

- `BEQ label` → ako je rezultat prethodne operacije nula (Zero bit = 1).
- `BNE label` → ako rezultat nije nula.
- `BGE label` → ako je veće ili jednako.
- `BLT label` → ako je manje.

if else struktura

-I

```

CMP r0, r1      ; usporedi r0 i r1
BGE fblock      ; ako je a >= b, idi na false blok
MOV r0, #5      ; x = 5
ADR r4, x
STR r0, [r4]    ; pohrani 5 u x
B after_if
fblock:
SUB r0, r0, r1   ; x = c - d
ADR r4, x
STR r0, [r4]
after_if:

```

petlje

jezik : C

```

for(i = 0, f = 0; i < N; i++)
    f = f + c[i] * x[i];

```

implementacija petlje : M A S

```

MOV r0, #0      ; i = 0
MOV r2, #0      ; f = 0
loop:
    LDR r4, [r3, r8] ; učitaj c[i]
    LDR r6, [r5, r8] ; učitaj x[i]
    MUL r4, r4, r6   ; c[i] * x[i]
    ADD r2, r2, r4    ; f += c[i] * x[i]
    ADD r8, r8, #4    ; povećaj offset
    ADD r0, r0, #1    ; i++
    CMP r0, r1        ; usporedi i s N
    BLT loop          ; ako je i < N, ponovi petlju

```

rad sa stogom

koristi BL (Branch with Link) instrukciju :

```

BL funkcija ; poziva funkciju i sprema povratnu adresu u r14 (LR - Link Register)

```

koristi (C, r MOV r15, r14 1 P 5 14 R. = =)

ene funkcije i rad sa stogom

đ

i stog za čuvanje povratnih adresa ugnijež enih funkcija

:

đ

```
STR r14, [r13, #-4]! ; sačuvaj povratnu adresu na stog
BL f2                ; pozovi f2
LDR r14, [r13], #4   ; vrati povratnu adresu sa stoga
MOV r15, r14         ; povratak iz funkcije
```

ključak

Z

Ovaj dokument detaljno pokriva osnovnih karakteristika i registara instrukcija kontrole toka i funkcijskih poziva . Za učenje je ključno:

organizaciju registara i memorije .

osnovne instrukcije R M A

se koristi grananje i petlje .

to se funkcije pozivaju i kako se koristi stog .