Sažetak PDF-a: Ugradbeni sustavi - ARM arhitektura

Dokument opisuje **ARM arhitekturu**, koja se temelji na **RISC (Reduced Instruction Set Computing)** konceptu i koristi se široko u **ugradbenim sustavima**. Ovdje su ključne informacije za učenje:

1. Osnovne značajke ARM arhitekture

- ARM ne proizvodi gotove procesore već daje **licence** proizvođačima.
- ARM arhitektura je **RISC** arhitektura, što znači da koristi **jednostavan skup instrukcija** i **load/store pristup**.
- **Assembly jezik ARM procesora** koristi čitljive instrukcije i oznake memorijskih lokacija.

2. Verzije ARM arhitekture i obitelji procesora

- **Cortex-A**: Za **aplikacije visokih performansi** (smartfoni, digitalni TV uređaji).
- **Cortex-R**: **Real-time** sustavi (ABS sustavi, napajanje vlakova).
- **Cortex-M**: **Mikrokontroleri** (pametna senzorska rješenja).
- **SecurCore**: **Sigurnosne aplikacije**.
- Ranije verzije: **ARM7, ARM9, ARM11**.

3. Memorijska organizacija i procesorska arhitektura

- **ARM može koristiti Von Neumann ili Harvard arhitekturu**.
- **Little-endian vs. Big-endian** organizacija memorije.
- **ARM koristi 32-bitne riječi**, koje se mogu dijeliti na 4 bajta.
- **Programski brojač (PC - r15) inkrementira se za 4 pri svakoj instrukciji**.

4. Rad s podacima u ARM arhitekturi

- **Load-store model**: Podaci se prvo učitavaju u registre prije obrade.
- **16 registara opće namjene** (r0-r15), pri čemu je **r15 PC registar**.
- **CPSR registar** pohranjuje informacije o statusu rezultata operacija:
    - **N** (negativan rezultat)
    - **Z** (nula rezultat)
    - **C** (prijenos/podlijevanje)
    - **V** (preljev)

5. Skup instrukcija ARM arhitekture

- **Aritmetičke operacije**: ADD, SUB, MUL, MLA.
- **Logičke operacije**: AND, ORR, EOR.
- **Pomicanje bitova**: LSL, LSR, ASR, RRX.
- **Usporedbe**: CMP (uspoređuje bez mijenjanja registara).
- **Prijenos podataka**:
    - **Iz memorije u registre**: LDR, STR.
    - **Indirektno adresiranje registara** omogućuje pohranu adresa u registre.

6. Kontrola toka izvršavanja

- **Grananje (B - Branch)**:
  - **Uvjetno grananje**: BNE (ako nisu jednaki), BEQ (ako su jednaki).
  - **Relativno grananje** koristi **offset** od PC registra.
- **Implementacija IF grananja**:

```
CMP r0, r1   ; usporedi a i b
BGE false_block  ; ako je a >= b, idi na false blok
MOV r0, #5   ; x = 5
B after_if   ; preskoči false blok
false_block: SUB r0, r0, r1  ; x = c - d
after_if:
```

- **Switch-case** se može optimizirati korištenjem **branch table-a**.

7. Implementacija petlji

- **For i while petlje koriste uvjetna grananja i base-plus-offset adresiranje**.
- **Primjer for petlje u C jeziku**:

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

  - U **ARM assembly** to se prevodi pomoću **LDR, MUL i ADD** instrukcija.

8. Pozivanje funkcija i rad sa stogom

- **BL (Branch and Link)** instrukcija se koristi za pozivanje funkcija.
- **Povratak iz funkcije koristi r14 (LR - Link Register)**:

```
BL funkcija  ; poziv funkcije
MOV r15, r14  ; povratak iz funkcije
```

- **Ugniježđene funkcije koriste stog za čuvanje povratnih adresa**.

Zaključak

Ovaj dokument pokriva ključne koncepte ARM arhitekture, memorijsku organizaciju, skup instrukcija, kontrolu toka i funkcijske pozive. Za učenje je ključno razumjeti:

1. **ARM arhitekturu i njezine obitelji procesora**.
2. **Osnovne instrukcije i rad s registrima**.
3. **Kako se koristi grananje i petlje u ARM assembly jeziku**.
4. **Kako se funkcije pozivaju i kako se koristi stog za povratne adrese**.