Sažetak PDF-a: ARM arhitektura – komentirano

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

---

1. Osnovne karakteristike ARM arhitekture

- **RISC (Reduced Instruction Set Computing)** arhitektura: koristi **jednostavan skup instrukcija**, bržu izvedbu, ali više instrukcija za istu funkcionalnost u usporedbi s CISC arhitekturama.
- **Load-store arhitektura**: operacije se izvode samo na registrima, a podaci iz memorije prvo se učitavaju u registre, obrađuju i zatim pohranjuju natrag.
- **Podaci u memoriji organizirani su u 32-bitne riječi** (4 bajta), a mogu se zapisivati **little-endian** ili **big-endian** načinom.

---

2. Organizacija registara u ARM arhitekturi

- **16 registara opće namjene (r0-r15)**:
  - **r0-r14**: opće namjene, koriste se za aritmetičke i logičke operacije.
  - **r15 (PC - Program Counter)**: sadrži adresu sljedeće instrukcije.
- **CPSR registar (Current Program Status Register)**:
  - **N** (negativan bit) – postavlja se ako je rezultat negativan.
  - **Z** (zero bit) – postavlja se ako je rezultat 0.
  - **C** (carry bit) – postavlja se pri preljevu rezultata.
  - **V** (overflow bit) – postavlja se ako dolazi do prekoračenja u izračunu.

---

3. Skup instrukcija ARM arhitekture

Aritmetičke operacije

- `ADD r0, r1, r2` → r0 = r1 + r2
- `SUB r0, r1, r2` → r0 = r1 - r2
- `MUL r0, r1, r2` → r0 = r1 * r2
- `RSB r0, r1, r2` → r0 = r2 - r1 (oduzimanje u obrnutom redoslijedu)

Logičke operacije

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

Pomicanje 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).

Usporedbe

- `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.

Prijenos podataka između 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.
- **Indirektno adresiranje**:
  - `LDR r0, [r1, #4]` → r0 = sadržaj memorijske lokacije na adresi (r1 + 4).
  - `LDR r0, [r1, -r2]` → r0 = sadržaj memorijske lokacije (r1 - r2).

---

4. Kontrola toka izvršavanja

Grananja

- `B label` → bezuvjetno grananje na adresu `label`.
- **Uvjetna grananja**:
  - `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

```
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

- **For petlja u C jeziku**:

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

- **ARM implementacija petlje**:

```
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
```

---

5. Pozivanje funkcija i rad sa stogom

- **Pozivanje funkcije koristi `BL (Branch and Link)` instrukciju**:

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

- **Povratak iz funkcije koristi `MOV r15, r14` (r15 = PC, r14 = LR)**.

Ugniježđene funkcije i rad sa stogom

- **ARM koristi stog za čuvanje povratnih adresa ugniježđenih funkcija**:

```
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
```

Zaključak

Ovaj dokument detaljno pokriva **ARM arhitekturu**, od **osnovnih karakteristika i registara** do **instrukcija, kontrole toka i funkcijskih poziva**. Za učenje je ključno:

1. **Razumjeti organizaciju registara i memorije**.
2. **Savladati osnovne ARM instrukcije**.
3. **Znati kako se koristi grananje i petlje**.
4. **Shvatiti kako se funkcije pozivaju i kako se koristi stog**.