Arhitektura micro RISC procesora i specifikacija asemblera za posmatrani procesor

Opis arhitekture

- 32-bit RISC procesor, značajno pojednostavljen za edukacione potrebe.
- Load/store RISC ortogonalna arhitektura.
- 32-bitni (virtuelni) adresni prostor, adresibilna jedinica bajt, little-endian format.
- troadresne instrukcije za celobrojnu aritmetiku (nema podrške za realne brojeve), označeno i neoznačeno poređenje i proširivanje.
- Dva režima izvršavanja: privilegovani (kernel, sistem) i korisnički režim rada.
- Ostatak specifikacije prikazuje samo dio arhitekture za korisnički mod (dio arhitekture vidljiv kompajleru i aplikativnom programeru).

Registri

- 16 x 32-bitnih opštenamjenskih registara: R0..R15
- 32-bitni programski brojač: PC
- 32-bitni pokazivač na vrh steka: SP; stek raste ka višim adresama, SP pokazuje na riječ na vrrhu steka.

Address Modes

- Neposredno adresiranje: *#konstantan_izraz*
- Registarsko direktno: R*i*
- Memorijsko direktno: *adresa; adresa* je konstantan izraz.
- Registarsko indirektno: [R*i*]
- Registarsko indirektno sa pomerajem: [R*i* + *offset*]; *offset* je konstantan iztaz.
- Registri koji se mogu koristiti u adresnim modovima su: R0..R15, PC, SP
- Korišćenje adresnih modova:
    - u instrukcijama za kontrolu toka: svi memorijski modovi (memorijsko direktno, registarsko indirektno i registarsko indirektno sa pomjerajem)
    - u Load/Store instrukcijama: svi (osim neposrednog za store instrukciju)
    - u stek instrukcijama, aritmetičkim i logičkim: registarsko direktno

- Format instrukcije:
  - registarsko direktno ili registarsko indirektno adresiranje:

31	23	20	15	10	5	2
Op code	Addr Mode	Reg0	Reg1	Reg2	Type	Unused

- neposredno, memorijsko direktno, registarsko indirektno sa pomerajem:
  - prva dupla reč (prvih 32 bita):

31	23	20	15	10	5	2
Op code	Addr Mode	Reg0	Reg1	Unused	Type	Unused

Second double-word (second 32 bits):

31
Constant/Address/Displacement

- Address Mode Codes:

neposredno	registarsko direktno	memorijsko direktno	registarsko indirektno	registarsko indirektno sa pomerajem
0b100	0b000	0b110	0b010	0b111

- Register codes:

R0..R15	SP	PC
0x0..0xf	0x10	0x11

- Data type codes:

dupla reč (32 bita)	reč (16 bitova) proširena nulama	reč (16 bitova) proširena znakom	bajt (8 bitova) proširen nulama	bajt (8 bitova) proširen znakom
0b000	0b001	0b101	0b011	0b111

Instrukcije

Instrukcije za kontrolu toka

- Adresa odredišne instrukcije se dobija kao adresa operand u memorijskim adresnim modovima.

•Format:

•koristeći registarsko direktno ili registarsko indirektno adresiranje:

31	23	20	15	10
Op code	Addr Mode	Reg0	Reg1/Unused	Unused

•koristeći memorijsko direktno adresiranje ili registarsko indirektno adresiranje sa pomerajem:

•Prva dupla reč (prvih 32 bita):

31	23	20	15	10
Op code	Addr Mode	Reg0	Reg1/Unused	Unused

•Druga dupla reč (drugih 32 bits):

31
Address/Displacement

•Instrukcije:

Instruction	*Op code*	*Address Modes*	*Comment*
INT op	0x00	Registarsko direktno (broj prekida)	Softversko generisanje prekida/ sistemski poziv; Reg0 sadrži broj prekida.
JMP op	0x02	memorijsko direktno, registarsko indirektno, registarsko indirektno sa ili bez pomeraja	Apsolutni ili relativni skok, u zavisnosti od načina adresiranja.
CALL op	0x03	-||-	Poziv potprograma. PC se smečta na stek.
RET	0x01	None	Povratak iz potprograma. Vrednost sa vrha steak se skida i smešta u PC
JZ reg1, op	0x04	Za op: memorijsko direktno, registarsko indirektno, registarsko indirektno sa ili bez pomeraja Za reg1: registarsko direktno	Ako je Reg1==0, skoči na op
JNZ reg1, op	0x05	-||-	Ako je Reg1!=0, skoči na op
JGZ reg1, op	0x06	-||-	Ako je Reg1>0 (označeni brojevi), skoči na op

JGEZ reg1, op	0x07	-||-	Ako je Reg1>=0 (označeni brojevi), skoči na op
JLZ reg1, op	0x08	-||-	Ako je Reg1<0 (označeni brojevi), skoči na op
JLEZ reg1, op	0x09	-||-	Ako je Reg1<=0 (označeni brojevi), skoči na op

Load/Store instrukcije

- Load: kopira podatak iz memorije u registar
- Store: kopira podatak iz registra u memoriju
- Load, veličine i tipovi operanada:
  - bajt, proširen nulama ili znakom; sufiks za mnemonik: UB (neoznačen) or SB (označen)
  - reč (16 bitova), proširen nulama ili znakom; sufiks za mnemonik: UW (neoznačena) or SW (označena)
  - dupla reč (32 bita); bez sufiksa
- Store, veličine i tipovi operanada:
  - bajt, bajt najmanjeg značaja iz registra; sufiks za mnemonik: B
  - reč (16 bitova), proširen nulama ili znakom; sufiks za mnemonik: W
  - dupla reč (32 bita); bez sufiksa
- Format:
- koristeći registarsko direktno ili registarsko indirektno adresiranje:

31	23	20	15	10	5	2
Op code	Addr Mode	Reg0	Reg1	Unused	Type	Unused

- koristeći memorijsko direktno adresiranje ili registarsko indirektno adresiranje sa pomerajem:
- Prva dupla reč (prvih 32 bita):

31	23	20	15	10	5	2
Op code	Addr Mode	Reg0/Unused	Reg1	Unused	Type	Unused

- Druga dupla reč (drugih 32 bita):

31
Address/Displacement

•Instrukcije:

Instruction	*Op code*	*Address Modes*	*Comment*
LOAD reg1, op	0x10	all	Load
STORE reg1, op	0x11	all except immediate	Store

Instrukcije za rad sa stekom

•Smeštanje sadržaja registra na stek i skidanje vrednosti sa vrha steak u registar

•Stek raste ka višim adresama, SP pokazuje na poslednji smešteni bajt

•Instrukcije menjaju SP implicitno

•Samo registarsko direktno adresiranje

•Na stek se uvek smešta i sa stek se uvek skida 32-bitna vrednost.

•Format:

31	23	20	15
Op code	Addr Mode = 0b000	Reg0	Unused

•Instrukcije:

Instruction	*Op code*	*Address Modes*	*Comment*
PUSH reg	0x20	Registarsko direktno	Push register
POP reg	0x21	Registarsko direktno	Pop register

Aritmetičke i logičke instrukcije

•Operacije nad 32-bitnim vrednostima (nisu moguci manji operandi)

•Troadresne operacije, koriste samo registarsko direktno adresiranje

•Označena celobrojna aritmetika

•Format:

31	23	20	15	10	5
Op code	Addr Mode = 0b000	Reg0	Reg1	Reg2	Unused

•Instrukcije:

Instruction	*Op code*	*Address Modes*	*Comment*
ADD reg0, reg1, reg2	0x30	Registarsko direktno	Reg0 = Reg1 + Reg2
SUB reg0, reg1, reg2	0x31	-‖-	Reg0 = Reg1 -Reg2
MUL reg0, reg1, reg2	0x32	-‖-	Reg0 = Reg1 * Reg2
DIV reg0, reg1, reg2	0x33	-‖-	Reg0 = Reg1 / Reg2
MOD reg0, reg1, reg2	0x34	-‖-	Reg0 = Reg1 % Reg2
AND reg0, reg1, reg2	0x35	-‖-	Reg0 = Reg1 &Reg2
OR reg0, reg1, reg2	0x36	-‖-	Reg0 = Reg1 |Reg2
XOR reg0, reg1, reg2	0x37	-‖-	Reg0 = Reg1 ^Reg2
NOT reg0, reg1	0x38	-‖-	Reg0 = ˜ Reg1
ASL reg0, reg1, reg2	0x39	-‖-	Reg0 = Reg1 <<Reg2
ASR reg0, reg1, reg2	0x3A	-‖-	Reg0 = Reg1 >>Reg2

Asembler

- Iskazi:
- instrukcije
- direktive (definisanje podataka i druge direktive)
- Format instrukcije:

labela: *op* *oper1, oper2, oper3* *;* *comment*

- *labela* može biti korišćena u konstantnim izrazima i izračunava se u apsolutnu adresu instrukcije ispred koje stoji (adresu podatka u slučaju da se nalazi ispred direktive za definisanje podataka).

Definisanje podataka

- Format definicije:

[label:] *def* *data-specifier [, ...]* *[;* *comment]*

- Moguće definicije:
  - DB: definiše bajt, svaka početna vrednost je veličine bajta;
  - DW: definiše reč, svaka početna vrednost je veličine 16 bitova;
  - DD: deviniše duplu reč, svaka dupla reč je veličine 32 bita.
- Specifikacija početnih vrednosti:

konstantan_izraz [DUP *konstantan_izraz* | ?]

- *konstantan_izraz:* literali, celobrojni aritmetički operatori (+, -, *, /), podizrazi sa zagradama

- *literal*: označen ceo broj u decimalnom, binarnom ili heksadecimalnom formatu sa prefiksima kao u jeziku C, kao i karakter po formatu kao u jeziku C.
- ? za nedefinisane početne vrednosti.
- DUP: prvi izraz definiše broj ponavljanja drugog izraza kao početne vrednosti, pri čemu su sva ponavljanja iste veličine kao i drugi izraz.

Direktive

- Definisanje simboličke konstante koja može biti korišćena u konstantnim izrazima:

symbol DEF constant-expression [; comment]

- Implicitno definisan simbol $ označava startnu adresu instrukcije u kojoj se koristi.
- Definisanje apsolutne adrese sledeće instrukcije (menja vrednost $ simbola u narednoj instrukciji):

ORG constant-expression [; comment]

- Definisanje segmenata:
  - .text[.broj] – sekcija sa programskim kodom
  - .data[.broj] – sekcija sa podacima koji imaju početne vrednosti
  - .rodata[.broj] – sekcija sa nepromenljivim podacima
  - .bss[.broj] – sekcija sa neinicijalizovanim podacima
- Iza naziva sekcije opciono može da se navede broj odvojen tačkom od osnovnog naziva sekcije. Osnovni naziv sekcije određuje tip sekcije, dok je sekcija sa brojem različita od ostalih sekcija.

Dodatna proširenja arhitekture sistema (potrebno za emulator)

- Prekidi:
  - mehanizam prekida je vektorisan, pri čemu IVT počinje od adrese 0 i zadrži 32 ulaza
  - u toku izvršavanja prekida nisu dozvoljeni ugneždeni prekidi
  - izvršavanjem instrukcije INT 0 se završava rad emulatora
  - ulaz 0 u IVT sadrži početnu vrednost SP registra
  - ulaz 3 sadrži adresu prekidne rutine koja se izvršava u slučaju bilo koje greške
  - ulaz 4 sadrži adresu prekidne rutine koju je potrebno izvršavati periodično sa periodom 0.1s realnog vremena

- ulaz 5 sadrži adresu prekidne rutine koja se izvršava svaki put kada je pritisnut neki taster emuliranog računara

- U memorijski adresni prostor su mapirana i dva registra, neposredno posle IVT. Prvi registar je mapiran na adresu 32 i predstavlja adresu registra podataka za ispis na standardni izlaz. Registar je veličine bajta. Vrednost upisana na ovu adresu predstavlja ASCII kod znaka koji treba ispisati na standardni izlaz. Drugi registar je mapiran na adresu 36 i predstavlja registar podataka ulazne periferije. Nakon što se dogodi prekid na ulazu 5, ovaj registar sadrži ASCII kod znaka koji odgovara pritisnutom tasteru. Vrednost se može čitati i više puta, sve dok se ne dogodi naredni prekid na ulazu 5, kada u tom registru postaje dostupna naredna vrednost. Na nivou emulatora obezbediti sinhronizaciju da se ne generiše novi prekid pre nego se pročita ranije postavljena vrednost

- Obzirom da je adresni prostor emulirane arhitekture 4GB, jasno je da nije moguće emulirati čitav adresni prostor, već je neophodno implementirati rešenje koje će obezbediti alokaciju dovoljne količine memorije. Količina alocirane memorije ne bi trebala da bude značajno veća od neophodne za izvršavanje programa.