1. PREZENTACIJA

PROCESOR

⇒ Ulaz, izlaz, memorija, put podataka, upravljačka jedinica

- Ulaz, izlaz → privuče/šalje podatke povezivanje
- memorija → sprema podat. bez rezultate
- put podataka → obavlja neke operacije privremeno pohrani podatke i međurez. prenosi podatke M-UJ
- upravljačka jedinica → upravlja radom PP, memorije, OI sklopova

⇒ fetch / decode / execute

⇓ može i istovremeno i preko memorije

⇒ apstrakcija: arhitektura skupa naredbi, mikroarhitektura

⇒ von Neumannova / Harvardska arh.

⇓ ↳ 2 veze - programska M-P-podatkovna M

usko grlo - jedna veza P-M

MEM

⇒ spojni putevi - SABIRNICE - adresna, podatkovna, upravljačka

⇒ adresni dekoder

⇒ mem riječ

* ⇒ STOG - LIFO - SP ⇒ registar (PUSH / POP)

ARH

⇒ STOGOVNA - stog unutar procesora, ali gubi se međurez. → puno pristupa memoriji

⇒ AKUMULATORSKA - jedan operand u reg ACC, drugi u mem; jednoadresna

⇒ REG/MEM - jedan operand u skupu reg opće namjene, drugi u mem, rez u reg ON

⇒ REG/REG - oba operanda -II- -II- -II-, LOAD/STORE, -I I-

SKUP NAREDBI: CISC jednostavnije prevodenje, štedi mem, ubrzanje rada
veliki br. naredbi i načina adresiranja, skupo

RISC manji skup jed. naredbi, mali broj načina adresiranja, protočna struktura, relativno jeftino

⇒ br. reg ⇒ potencija broja 2 (R0-R7)

- Širina: 32 bita
- podatkovna sabirnica: 32 bita
- širina mem riječi: 8 bita (možemo pročitati 4 adjedna)

⇒ PC - reg u kojem se nalazi adresa iduće naredbe

NAČINI ADRESIRANJA

procesorska: registarsko
neposredno (kao broj, ne adresa; može i labela)
apsolutno (broj kao adresa ili labela)
relativno (kod JR - broj nije adresa)
registarsko indirektno sa ili bez odmaka (registar + odmak u zagradi)
implicitno (kod PUSH i POP - samo registar jer se zna da su podatci u vrhu stoga)

asemblerska

NAREDBE + STROJNI KOD

aritmetičko-logičke

bitovi: ADD, SUB, AND, XOR, OR, ADC, SBC
- 31-27 → operacijski kod
- 26 - ako je 0 → 2 reg; ako je 1 → reg i broj
- 25-23 dest
- 22-20 STC1
- 19-17 STC2 ili 19-0 broj (predznačno se proširuje) } služe za višestruku preciznost (ako zbrajamo 2 64b broja, niže bitove zbrajamo normalno, a više pomoću ADC)

CMP - SUB, ali bez rezultata → samo postavi zastavice

SHL, SHR } logički pomak (-puni s 0)

ASHR → puni s vodećim bitom

ROTL, ROTR } rotacije

registarske

MOVE → 31-27 OK, 25-23 dest
- 22-20 - za 000 - Ri → Ri => 26 ∪ 0 - 19-17 STC, } 19-0 podatak ako je 26 ∪ 1 - umjesto
- za 001 - Ri → SR => 26 ∪ 0 - 19-17 STC, } (predznačno se proširuje)
- za 010 - SR → Ri => 26 i 19-0 se ne koriste

memorijske

od LOAD, ORE, PUSH i P uzimaju 2 ta su automatski stavljena u 0 adr. mora bit

LOAD(B,H), STORE(B,H) } 31-27 OK, 25-23 dest/STC, 26 ∪ 0 → 19-0 adr, 26 ∪ 1 → 22-20 adr re
*B - LOAD/STORE jednog bita; H - LS jedne poluriječi (4b) → 19-0 odnos

PUSH, POP } 31-27 OK, 25-23 STC/dest

živa s 4 1 H zadnji bit)

R7 => SP (samo ako se koristi stog) - pokazuje na zadnji pod. na stogu (last in
*Uvijek +4 (cijela riječ)

upravljačke

JP - 31-27 OK, 25-22 uvjet, 19-0 adresa ako je 26 ∪ 1; 19-17 adr reg ako je 26 ∪
*JP ili JP_uvjet

JR - 31-27 OK, 25-22 uvjet, 19-0 odmak od trenutne pozicije

CALL - za poziv potprograma - sve isto kao kod JP osim 31-27(OK), sprema povratnu adresu u PC

RET - uzme pov. adr iz PC i skače na nju (povratak iz potprog.) i uto

RETI iut = 1, i/u = 0 } prebacuje prijenos

RETN iut = 1, i/u = 1

HALT 31-27 OK, 25-22 uvjet - zaustavljanje rada procesora

31-27 OK
25-22 uvjet
1 - i/u
0 - iut

FIZIČKA MEM ≤ LOGIČKE MEM

=> zbog predznačnog proširivanja koristi se samo 1/4096 MB

(0000 0000 - 0007 FFFF) ∧ (FFF8 0000 - FFFF FFFF) - viših 13 bitova mora

2. PREZENTACIJA

asemblери - prevode programe u strojni kod

niskorazinski jezik - jezik niže razine, prilagođen procesoru

apsolutni i premjestivi tip programa

↓ sve određeno ↓ odrediti početnu adresu

⇒ **ATLAS** - simulator računala na niskoj razini

↳ asemblerski prevoditelj: **COVAS** (dvoprolazni)

⇒ labele - imena adresa: brže, bolje i lakše programiranje

↳ COVAS razlikuje prvih 20 znakova (1HE12345612345612345 = 1HE12345612345612345 1HE)

↳ koristi se samo jednom

pseudonaredbe

'ORG - zadaje adresu, automatski ju postavlja na prvu djeljivu s 4

'DW - izravan upis podatka u memoriju, ne postavlja na adr. djeljivu s 4

'EQU - daje vrijednost labeli (↳ 8 bita), ne zauzima memoriju

'DS - zauzima mem. s određenom početnom vrijednošću (inicijalno 0), ne postavlja na adr. djeljivu

'END - prekid prevođenja

'BASE - definiranje baze brojevnog sustava (B, O, D, H) od trenutne pozicije do iduće naredbe

pisanje brojeva → inicijalno HEKSADEKADSKI

↳ format za promjenu baze jednog broja: %D 12 (ili B, O, H)

→ brojevi započinju znamenkom! ABC = 0ABC (labela) ↳ 274_{10}

DW / DH / DB → definiranje podatka od 8/4/1 bita - na adr. djeljivu s 4!

→ češće se koristi nego 'DW, za razliku od 'DW, podatak može biti i labela

Uspoređivanje brojeva - CMP najbolji izbor (postavi zastavice, ne zauzima dodatno mj

Rad s bitovima - osnovne operacije: set, reset, komplement, test

- NIŽI / VIŠI BITOVI - najniži sbroz desno, najviši skroz lijevo
- NIŽI / VIŠI BAJT - 1.-8. bita / 9.-15. bita

MASKA - niz 0 i 1 - koristi nam za osnovne operacije

▽ 0	set	reset	komplement	ostaje isto
AND		0		1
OR	1			0
XOR			1	0

PAZI! - mora se napisati CIJELA konstanta (32b) - asembler prepoznaje predznačno proširenje

* pravi primjere ispitivanja bitova !!!

UPIS - na početak ili kraj - shift!

mijenjanje - pomoću maske

PREBRAJANJE - rotacijama! (bit po bit)

višestruka preciznost

- koristi se više registara
- ADC, SBC ↳ bouplement zastavice C
  ↳ zastavica C
- AND, OR, XOR uobičajeno
- pomaci i rotacije → pomaci, samo što se zadnji bit dovodi na mjesto prvog ili obrn
  ↳ svaku iduću riječ punimo izlazom iz prethodne
- proširivanje: iz višestruke u jednostruku: gube se podatci osim ako su svi viši bitovi 0 za NBC ili 0/1 za 2'k

  iz jednostruke u višestruku: za NBC punimo s 0

  za 2'k punimo s vodećim bitom

POTPROGRAMI → INICIJALIZIRATI SP!! (MOVE 10000, SP)

→ stog počinje na poziciji FFFF prema nižim adresama

prijenos podataka: registrom - programer stavlja podatke u reg i u potprogramu ih koristi; brzo, jednostavno, ograničen broj argumenata, nemoguća rekurzija

fiksnom lok. - programer daje vrijednost neke lokacijama ih koristi u potprog., sporo, jednostavno, ograničen broj argumenata, nemoguća rekurzija

stogom - CALL stavlja pov. adr. na stog, PUSH i POP služe za prijenos podataka iz glavnog prog. u potprog., RET uzima povratnu adresu sa stoga

⇒ glavni prog. stavlja parametri reg. na stog, CALL stavlja pov. adr., potp. stavlja reg. koje će koristiti, pomiče po stogu: registarsko indirektno adresiranje s odmakom obnavljaju se korišteni registri → za povratak iz potprograma koristi se prijenos registrom (vraćanje rezultata), RET vraća u glavni prog., gp. oblaya parametri reg.

+ lokalne varijable - samo u potprog. - čuvaju se na stogu

REKURZIJA → potprog. poziva samog sebe

- kao da je potprog. više puta aktiviran => prijenos parametara isključivo stogom
- vraćanje povratne vrijednosti može preko registra (uvijek je samo jedna)
- primjeri!!

MAKRONAREDBE → ista namjena kao potprog.; zbog samo 2 prolaza nemoguće u ATLAS-u

→ razlika u načinu izvedbe: asembler prevodi makrou. u obične naredbe

→ samo troprolazni i četverostruki asemblleri

↳ def. mora biti ispred poziva ↳ def. bilo gdje

→ parametri makronaredbi

4 { PRVI PROLAZ: zapamti ime i tijelo, ne mijenja ostalo
DRUGI -11- : mijenja sve pozive MN njihovim tijelima
TREĆI -11- : } kao 1. i 2. prolaz dvoprolaznog asembliranja
ČETVRTI -11- :

3 { PRVI PROLAZ: zapamti ime i tijelo, poziv zamjenjuje s tijelom
DRUGI -11- : } kao 1. i 2. prolaz dvoprolaznog asembliranja
TREĆI -11- :

3. PREZENTACIJA

PRIKLJUČCI → služe za spajanje na sabirnicu
- adresni, podatkovni, upravljački
  - ↓ postavljaju adr. na sabirnicu
  - ↓ prijenos podataka (P-M, P-VJ)
  - ↓ sinkronizacija rada
- aktivni visko: $\overline{IHE1}$
- D - " - visoko: IHE2
  + 2 stanja: $\overline{IHE1}$/IHE2

} rade uglavnom na brid signala takta

- samostalni ili u sklopini

sučelje: ulazni
izlazni
dvosmjerni - samo jedna komponenta upravlja sabirnicom (load i store se ne mogu istodobno)
- sklop s 3 stanja

Open collector priključci - svi priključci na jednoj sabirnici, svi upravljaju

PRIKLJUČCI FRISCA-

Vcc, GND - napajanje

CLOCK - sinkronizacija - signal takta

iz ADR - 32-bitni skup adresnih p.

iz/ul DATA - " - " - podatkovnih p.

iz $\overline{READ}$ - čitanje iz memorije
iz $\overline{WRITE}$ - pisanje u memoriju
} aktivni visko

iz SIZE - 2-bitni skup koji zadaje širinu podatka (00-0, 01-8b, 10-16b, 11-32b)

ul $\overline{WAIT}$ - služi za produljenje ciklusa

ul $\overline{RESET}$ - dovodi procesor u početno stanje

---

ul $\overline{INT}$ - 4 priključka - zahtjev za prekid

iz $\overline{IACK}$ - potvrda zahtjeva za prekid

ul $\overline{BREQ}$ - DMA traži upravljanje sabirnicom

iz $\overline{BACK}$ - potvrda zahtjeva DMA-jedinice

SABIRNICE - memorijska, ulazno-izlazna, specijalne namjene

↓ povezuje P i M
- mala duljina
- velika brzina
- prilagođena brzini mem.

↓ povezuje U-I i P
- velika duljina
- manja brzina
- prilagodljiva raz. brzinama

SPAJANJE: zajedničke mem/U-I sabirnice pomoću međusklopa

SABIRNIČKI PROTOKOL - redosljed svih koraka u komunikaciji
- transakcija - slijed koraka za operaciju
↳ REQUEST/RESPONSE

- sinkroni → clock
  - jednostavno, brzo, često kad svi uređaji imaju istu brzinu
    ↳ memorijske
- asinkroni → bez clock-a ⇒ handshaking protocol
  ↳ idući korak tek nakon potvrde o izvršenju trenutnog
  - složenije, sporije, velike duljine, bolje prilagođenje raz. brzinama
    ↳ U-I sabirnice

- clock je u procesoru - ne koristi se za sve mem sabirnice, a koristi se i za neke korake asinkronih sabirnica

4. PREZENTACIJA

* fetch / decode / execute

PROTOČNA STRUKTURA - svaka razina - jedna faza
- kao pokretna traka
- UBRZANJE: sa $M \cdot N$ na $(M-1)+N$ koraka ▽ o
  (za veliki M: $(M-1)+N \approx M \Rightarrow \frac{M \cdot N}{M} = N$)
- brzina strukture ovisi o brzini najsporije faze

FRISC: fetch / execute
↓ ↳ izvodenje AL operacija, spremanje rez.
dohvat naredbe, dekodiranje, dohvat operanada

naredbe: DVOCIKLUSNE - memorijske, upravljačke
JEDNOCIKLUSNE

HAZARDI

→ **strukturni** - procesor ne može obaviti sve faze u protočnoj strukturi istodobno
↳ Von Neumann?
- RJEŠENJE: odgoda izvodenja - bubble
  - kad uzrok nestane, procesor nastavlja rad
- FRISC: LOAD, STORE, PUSH, POP - memorijske
  ↳ zbog V.N.arh. ima samo 1 pristup mem. - DVOCIKLUSNE NAREDBE
  - nakon prepoznavanja naredbe deaktivira se fetch - bubble
  - nakon izvodenja naredbe fetch se ponovo aktivira, bubble prelazi u fazu execute

→ **upravljački** - naredba koja je u strukturi uputa koja se treba izvesti
- kod JP i JR - nakon fetch faze JP/JR slijedi execute, a u fetch ulazi prva iduća naredba (ona ispod JP/JR)
- hazard grananja
- kada je uvjet skoka istinit, naredba JP/JR je dvociklusna?
- FRISC: JP i JR su zbog jednostavnosti uvijek DVOCIKLUSNE

* IZVOĐENJE NIZA NAREDBI: $T_{izvodenja} = 2u + 1(+1)$ → dodaje se samo ako je zadnja naredba jednociklusna
u - broj dvociklusnih; 1 - broj jednociklusnih

→ **podatkovni** - naredba se ne može izvesti jer podatci još nisu u spremniku
- ne postoji kod FRISC-a

5. PREZENTACIJA

POVEZIVANJE RAČUNALA S OKOLINOM

- uređaji se na sabirnicu procesora spajaju preko ulaza/izlaza (međusklopova)
  ↳ procesor ne vidi uređaj - komunicira samo s UI jedinicom

UIj. PO SMJERU: ulazne, izlazne, dvosmjerne

UIj. PO NAMJENI: za prijenos pod., brojanje impulsa, mjerenje vremena, generiranje impulsa, AD i DA pretvorbu, spec. namjenu, ...

načini adresiranja: memorijsko UI preslikavanje - mem i UIj. dijele isti adr. prost
↳ sabirničbi protokoli jednaki za UIj. i mem
izdvojeno UI adresiranje - razdvojeni adresni prostor
↳ pr. mem: LOAD/STORE; UI: IN/OUT
↳ posebni priključci $\overline{MEM}$/IO ili MEMRQ i IORQ
jednostavnije, manje priključaka
↳ ne smanjuje se mem. prostor, efikasnija komunikacija

FRISC: 0000-FFFF; UIj. FFFF0000-FFFF FFFF

- brzina UIj. ≠ brzina procesora => potrebna sinkronizacija

=> PROGRAMSKI PRIJENOS: prijenos obavlja procesor - sporiji
: prijenos podataka usporava izvođenje ostalih poslova
: bezuvjetni, uvjetni, prekidni

=> SKLOPOVSKI PRIJENOS: prijenos obavlja spec. jedinica: DMA kontroler
: brzo, ali svejedno usporava izvođenje programa
: izravni pristup memoriji: DMA

BEZUVJETNI PRIJENOS: najjednostavniji - ne provjerava je li UI spremna - bez sinkronizacije
- najbrži, najjednostavniji
- prijenos je jedna naredba
  NEDOSTATAK: mogućnost da UI nije spremna
- koristi se kada je sinkronizacija nepotrebna

UVJETNI PRIJENOS: uvijek provjerava je li UI spremna
  NEDOSTATAK: gubi se vrijeme na čekanju
- složeniji: bistabil stanja, sinkronizacijski priključci
- koristi se kada je bitna sinkronizacija
- dob je VJ spremna za komunikaciju s procesorom, a -||- -||- -||- -||- vanjskim procesom i obrnuto
- bistabil za sink.: VJ postavlja bistabil kad je spremna, automatski reset nakon prijenosa
  → brisanjem bistabila onemogućuje se komunikaciju vanjskim procesom
- građa UIj.: 2×32b - prva lokacija za čitanje/pisanje
  - druga -||- za pristup bistabilu
  - 8 adresa - početna određena adresnim dekoderom (CS, ADR2, READ/WRITE)

PREKIDNI PRIJENOS - UIj samostalno dojavljuje spremnost zahtjevom za prekidom
- bez gubitka podataka i tolikog gubitka vrem

prekidni sustav FRISC-a → inicijalno zabranjuju, sva 3 isti prioritet
INT0 - INT2 => maskirajući prekidi - zahtjev se može zabran
INT3 => nemaskirajući - " - - - " - se ne može - " -
↳ viši prioritet
IACK => potvrda prihvata nemask. prekida

SR => 8 bitni => 7 → GIE
6-4 → EINT → prekidne zastavice
3-0 → zastavice
*10-8 → INT - nepostojeći (samo u MOVE SR, Ri)
+ IIF - nije u SR-u - kod nemask.p. je u 0, inače je u 1 (čim se prihvati INT3 je u 0)

- ne prihvaćaju se novi zahtjevi dok je u

ISPITIVANJE PREKIDA: na padajući brid CLOCK-a
: ovisi o trenutnom stanju zastavica
① IIF = 0 - ne prihvaća se
② IIF = 1, INT3 prisutan - prihvaća se
③ IIF = 1, nema INT3 - maskirajući
④ GIE = 0 - mask. se ne prihvaćaju
⑤ GIE = 1 - mask. se prihvaćaju ako postoje

prihvaćanje nemask.p.: aktivira se IACK, briše se IIF
spremi se PC na stog
deaktivira se IACK
skok u prekidni potprogram
→ dojavljuje se sklopovski

prihvaćanje mask.p.: briše se GIE
spremi se PC na stog
dohvat adrese prekidnog potp., i skok u prekidni potp. ((8) → PC)
→ dojavljuje se programski

za nemask.: RETN → IIF = 1
za mask.: RET I → GIE = 1

OSNOVNA GRAĐA PREKIDNE UIj. - aktivna → sama postavlja zahtjev za prekid
→ može zabraniti prekid
- bistabil u 1 → zahtjev
- bistabil u 0 → briše zahtjev, ali ne ovogućava komunikaciju s vanjskim procesom

* za VJ koja se spaja na INT3 IACK briše bistabil?

-4 32-bitne lokacije: ① čitanje/pisanje podatka
② pristup bistabilu
③ upisom dojavljuje da je prekid ob
④ upravlja: 1 dozvoljava, a 0 zabra