Centralna jedinica za obradu
U prethodnom dijelu pažnja je bila posvećena pohranjivanju podataka i organizaciji same memorije u računalu. Jednom kada su podaci pohranjeni, stroj mora moći manipulirati sa tim podacima u skladu sa naredbama algoritma. Stroj, dakle mora imati ugrađenu sposobnost:

- izvođenja operacija i
- koordiniranja redoslijedom izvođenja operacija

Ovi zadaci se izvode u CPU, pa je glavna tema ovog dijela usko vezana uz način funkcioniranja centralne jedinice za obradu podataka (central processing unit – CPU).

ARHITEKTURA RAČUNALA

Dijelovi računala koji izvode operacije nad podacima (primjerice zbrajanja i oduzimanja nisu direktno vezani na memorijsku strukturu. Ti sklopovi čine zaseban funkcionalni dio računala - CPU (ili obično procesor), sastavljen od dva osnovna dijela: aritmetičko logičke jedinice u kojoj su sklopovi za manipulaciju sa podacima i upravljačke jedinice koja koordinira aktivnostima unutar računala. Za privremeno čuvanje podataka CPU koristi registre tzv. opće i posebne namjene. Registri opće namjene služe kao privremeni pričuvnici podataka koji se obrađuju unutar CPU. U ovim registrima čuvaju se vrijednosti koje ulaze u aritmetičko logičku jedinicu i čuvaju se rezultati koji se iz AL jedinice tamo pošalju. Da bi se izvela operacija nad podacima iz RAM-a, upravljačka jedinica mora:

1. voditi računa o tome da odgovarajući podaci dođu u registre opće namjene
2. informirati AL jedinicu u kojim registrima su podaci
3. aktivirati odgovarajuće sklopove unutar AL jedinice i
4. uputiti AL jedinicu gdje će pohraniti rezultat

Da bi mogli izmijenjivati podatke, CPU i RAM povezani su skupom žica – BUSom. Preko BUSa, CPU može dohvatiti ili pročitati podatke iz RAM-a proslijeđivanjem adrese odgovarajuće memorijske lokacije i signala za čitanje. Na isti se način podaci mogu upisati u memoriju postavljanjem na BUS adrese lokacije odredišta, postavljanjem podatka i signala za pisanje. Slijedom ovih uputa, zadatak zbrajanja dvaju pohranjenih vrijednosti iz RAM-a uključuje puno više od provođenja same operacije zbrajanja. Proces uključuje koordiniranu akciju upravljačke jedinice koja upravlja prijenosom informacija između radne memorije i registara unutar CPU-a i aritmetičko logičke jedinice koja izvodi operacije zbrajanja u trenutku kada joj upravljačka jedinica da znak da ih izvede. Cjelokupan postupak zbrajanja dvaju vrijednosti pohranjenih u memoriju, mogao bi se predstaviti procedurom sastavljenom od 5 koraka:

Korak 1: Uzmi prvu vrijednost koja se zbraja i pohrani je u registar
Korak 2: Uzmi drugu vrijednost koja se zbraja i pohrani je u drugi registar
Korak 3: Aktiviraj sklopove za zbrajanje sa vrijednostima koje su pohranjene u registrima tijekom prva dva koraka.
Korak 4: Rezultat operacije pohrani u memoriju.
Korak 5: Stani.

Ukratko, podaci se moraju prenijeti iz memorije u CPU, a rezultat se na kraju ponovno pohranjuje u memoriju. Starija računala nisu nudila fleksibilnost koja se dobila ovakvim

organiziranjem CPUa. Sklopovi koji su upravljali i izvodili operacije bili su sastavni dijelovi stroja, ni na koji način organizirani u posebnu funkcionalnu jedinicu. Drugačije funkcioniranje stroja, od onog za koje je bilo konstruirano, moglo se, u najboljem slučaju, dobiti prespajanjem dijelova CPUa preko upravljačkog pulta. Do bitog pomaka prema fleksibilnijem računalu došlo se sa arhitekturom koja je nudila mogućnost da se program, isto kao i podaci, pohrani u radnu memoriju. Upravo je John von Neumann-u pripala sva zasluga za definiranje arhitekture ovakvog tipa iako je, i prije nego što je on svoj rad objavio, Eckert-ov stroj izgrađen na tim načelima, uspješno funkcionirao. U njegovom je stroju, baš kao i u današnjim računalima, upravljačka jedinica dohvaćala program iz memorije, dekodirala instrukcije i izvodila ih, a namjena računala mijenjala se u skladu sa pohranjenim programom.

STROJNI JEZIK

Da bi se realiziralo računalo koje radi u skladu sa pohranjenim programom, CPU mora prepoznavati instrukcije kodirane nizom bitova. Ovaj skup instrukcija kodiranih binarnim nizovima predstavljaju strojni jezik.
Instrukcijski skup Možda iznenađujuće zvuči činjenica da je tipičan instrukcijski skup kojeg CPU može dekodirati i izvoditi jako malen. Kada i ako je dobro definiran, dodavanjem novih instrukcija ne povećavaju se teoretske mogućnosti stroja nego ga se eventualno prilagođava tekućim potrebama. Različiti pristupi u projektiranju instrukcijskog skupa rezultirali su različitim karakteristikama arhitektura današnjih procesora: RISC i CISC. Argumenti koji idu u prilog RISC arhitekturi (reduced instruction set computer) je efikasnost i brzina izvođenja velikog broja jednostavnih instrukcija. S druge strane, zagovornici CISC arhitekture (complex instruction set computer) smatraju da složeniji instrukcijski skup pruža veće mogućnosti jer će se sa jednom instrukcijom riješiti veći dio zadatka, makar neke od tih instrukcija bile redundantne. Iako se danas efikasnijim smatraju RISC procesori, u vrijeme kada je nastala, CISC arhitektura imala je opravdano puno zagovornika. Složeniji instrukcijski skup zahtijevao je manji memorijski prostor, što je tada predstavljalo značajnu uštedu zbog skupoće memorije. S vremenom, cijena memorije je padala, a istraživanja su pokazala da se tijekom 90% vremena u kojem se izvodi program, koristi samo 10% naredbi iz instrukcijskog skupa CISC procesora, dok je kod RISC procesora iskoristivost daleko veća. Time se polazna pretpostavka koja je išla u prilog uštedi memorijskog prostora pokazala potpuno pogrešnom. Danas na tržištu postoje procesori temeljeni i na jednoj i na drugoj arhitekturi. Pentiumovi procesori, koje je razvio Intel, tipični su primjeri CISC arhitekture, za razliku od PowerPC serije procesora koje su razvili AppleComputer, IBM i Motorola i koji su tipični predstavnici RISC arhitekture. Općenito. Instrukcije procesora mogu se svrstati u tri osnovne kategorije:

- instrukcije za prijenos podataka,
- aritmetičko-logičke instrukcije i
- upravljačke instrukcije.

Instrukcije za prijenos podataka:
U ovu grupu spadaju instrukcije koje zahtijevaju pomicanje podataka sa jednog na drugo mjesto. U gornjem primjeru koraci 1,2 i 4 spadali bi u ovu grupu. Instrukcije pomicanja kopiraju (a ne premještaju) vrijednosti iz radne memorije u odgovarajuće registre procesora. Pri tom, koristi se izraz LOAD kojim se naznačava da je riječ o upravo toj akciji, a za obrnutu proceduru pohranjivanja vrijednosti iz registara u radnu memoriju, uobičajeni je termin STORE. U spomenutom primjeru, instrukcije 1 i 2 primjeri su LOAD operacija, a istrukcija u 4. koraku naznačava STORE operaciju. U ovu kategoriju spadale bi i instrukcije kojima podatke iz radne memorije pohranjujemo na neku od vanjskih jedinica i zovemo ih I/O

instrukcijama. Zbog njihovih karakteristika, ponekad ih se svrstava i u zasebnu kategoriju.

Aritmetičko logičke instrukcije:

U ovu grupu spadaju instrukcije koje od upravljačke jedinice traže pokretanje aktivnosti unutar aritmetičko logičke jedinice. I samo ime za AL jedinicu sugerira da ova može osim uobičajenih aritmetičkih, obaviti i logičke operacije. U te spadaju, primjerice logičke AND, OR ili XOR operacije koje se često koriste za izmjenu samo nekih vrijednosti bitova u podacima pohranjenima u registre opće namjene. Druga skupina logičkih operacija omogućava pomicanje niza vrijednosti unutar registra u lijevo ili u desno (SHIFT i ROTATE).

Upravljačke instrukcije:

U ovu grupu spadaju instrukcije koje ne manipuliraju sa podacima nego usmjeravaju izvođenje programa. Krajnje jednostavni primjer instrukcija iz ove skupine, predstavljao bi 5. korak iz primjera – STOP. Osim te, u istu grupu spada i operacija JUMP, koja će vjerojatno više zakomplicirati program. Na donjoj slici predstavljen je algoritam kojim možemo dijeliti vrijednosti pohranjene u memoriji, uz pomoć spomenutih operacija.

1. LOAD registar sa vrijednošću iz memorije
2. LOAD drugi registar sa vrijednošću sa druge lokacije
3. Ako je druga vrijednost 0, JUMP na korak 6.
4. Podijeli sadržaj prvog sa drugim registrom i rezultat odloži u treći registar
5. STORE sadržaj trećeg registra u memoriju
6. STOP.

Ilustrativni primjer strojnog jezika

Promotrimo kako su kodirane instrukcije na jednom tipičnom stroju koji koristi 16 registara opće namjene i ima 256 adresa radne memorije na kojima se može pohraniti byte podatka. Zbog točnosti prikaza, neka su registri označeni vrijednostima od 0 do 15, a vrijednosti memorijskih adresa od 0 do 255 i to predstavljeno odgovarajućim binarnim ekvivalentima i heksadecimalnim zapisom. Kodirana strojna instrukcija ima dva tipična dijela: tzv. polje op-koda (operacijski kod) i polje operanda. Niz bitova u polju operacijskog koda predstavlja neku od instrukcija iz instrukcijskog skupa, a niz bitova u polju operanda indicira detalje potrebne za izvođenje instrukcije, primjerice koji se registar opće namjene koristi za podatak sa određene lokacije u memoriji. Cijeli instrukcijski skup ovog ilustrativnog primjera predstavljen je sa 12 osnovnih instrukcija. Svaka instrukcija kodira se sa 16 bitova (predstavljenih sa 4 heksadecimalne znamenke) na način predstavljen tablicom:

OP-CODE	OPERAND	OPIS
1	RXY	LOAD registar R sa uzorkom bitova na memorijskoj adresi XY Primjer: 14A3: Napuni registar 4 sa sadržajem memorijske lokacije A3
2	RXY	LOAD registar R sa uzorkom bitova XY
3	RXY	SAVE u registar R uzorak bitova XY
4	0RS	ADD uzorke bitova u registrima S i T predstavljenih dvojnim komplementom i rezultat pohrani u registar R
5	RST	ADD uzorke bitova u registrima S i T predstavljenih dvojnim komplementom i rezultat

		pohrani u registar R
6	RST	ADD uzorke bitova u registrima S i T predstavljenih plivajućim zarezom i rezultat pohrani u registar R
7	RST	OR uzorke bitova u registrima S i T i rezultat pohrani u registar R
8	RST	AND uzorke bitova u registrima S i T i rezultat pohrani u registar R
9	RST	XOR uzorke bitova u registrima S i T i rezultat pohrani u registar R
A	R0X	ROTATE uzorke bitova u registru R za X mjesta u desno
B	RXY	JUMP na naredbu pohranjenu na memorijskoj adresi XY ako je sadržaj R jednak sadržaju registra 0
C	000	STANI

Operacijski kod predstavljen je sa prva 4 bita, tj sa prvom heksadecimalnom znamenkom. Iz tablice je vidljivo da, primjerice, naredba koja počinje sa 3 označava operaciju pohranjivanja, a ona označena slovom «A» označava instrukciju ROTATE. Polje operanda svake operacije u ovom zamišljenom procesoru sastavljeno je od 3 heksadecimalne znamenke i u svim slučajevima (osim za naredbu HALT koja je sama po sebi jasna) predstavlja način izvođenja operacije predstavljene kodom. Primjer: 35A7 znači da treba pohraniti sadržaj registra 5 na memorijsku lokaciju sa adresom A7. Isto tako vrijedno je uočiti razliku u funkcioniranju LOAD operacija. Operacija «1» puni registar sa sadržajem neke memorijske lokacije, a operacija «2» puni registar sa uzorkom bitova predstavljenim heksadecimalnim znamenkama. I za operaciju zbrajanja također postoje dva načina korištenja: koristimo li je za zbrajanje cijelih, odnosno za zbrajanje znamenaka predstavljenih pomičnim zarezom.

Kodirane instrikcije predstavljene PROCEDUROM1 u skladu sa tablicom mogle bi imati oblik:

156C - napuni registar 5 sa sadržajem memorijske lokacije 6C
166D - napuni registar 6 sa sadržajem memorijske lokacije 6D
5056 - zbroji cjelobrojne vrijednosti u registrima 5 i 6 i pohrani u registar 0
306E - pohrani sadržaj registra 0 na memorijsku lokaciju sa adresom 6E
C000 - stani

Izvođenje programa

Računalo slijedi program pohranjen u memoriji kopiranjem instrukcija iz memorije u upravljačku jedinicu. Kada je naredba u upravljačkoj jedinici, svaka se instrukcija dekodira i pokreću se odgovarajuće akcije u skladu sa značenjem instrukcije. Red kojim se instrukcije dekodiraju odgovara redoslijedu kojim su pohranjene u memoriju, osim ako drugačije nije specificirano, primjerice JUMP naredbom. Da bi se razumjelo kako se izvodi program, nužno je izbliza promotriti funkcioniranje upravljačke jedinice unutar CPUa. Unutar te jedinice nalaze se dva registra posebne namjene:

programsko brojilo i
instrukcijski registar.
Programsko brojilo sadrži adresu memorijske lokacije sa slijedećom naredbom i služi kao oznaka za mjesto do kojeg se došlo u izvođenju programa. Upravljačka jedinica izvodi instrukciju tijekom u "strojnog ciklusa" sastavljenog od tri koraka tijekom kojih se: DOHVAĆA, DEKODIRA i IZVODI instrukcija. Kod dohvaćanja, upravljačka jedinica proslijeđuje (kopira) instrukciju sa memorijske lokacije naznačene programskim brojilom u instrukcijski registar. Kako je svaka instrukcija duga 2 byta, postupak dohvaćanja podrazumijeva prebacivanje podataka sa dvije memorijske lokacije u instrukcijski registar. Pri dekodiranju instrukcija se razbija na sastavne dijelove: op-code i polje operanda. U skladu sa op-codom definira se vrijednost programskog brojila i inicijaliziraju se sklopovi koji sudjeluju u izvođenju operacije čime se otvara put za konačno izvođenje same operacije. Time jedan strojni ciklus završava. Interesantni su primjeri instrukcija B258 i B058. U prvom slučaju riječ je o uvjetnom skoku: ako sadržaj registra 2 i 0 nije jednak, izvođenje se zaustavlja, a ukoliko je, onda programsko brojilo poprima vrijednost 58 i izvođenje se nastavlja. U drugom slučaju uspoređuje se vrijednost registra 0 sa samim sobom. Kako su te dvije vrijednosti uvijek iste, naredba predstavlja bezuvjetni skok na instrukciju pohranjenoj na adresi 58.

Instrukcije varijabilne dužine

Zbog pojednostavljenja, strojni jezik opisan tablicom koristi fiksnu dužinu za sve instrukcije. Dakle, da bi se dobavila instrukcija, CPU uvijek dobavlja sadržaj sa dviju susjednih lokacija i uvećava sadržaj programskog brojila za 2. U realnim sustavima, strojni jezici uglavnom koriste instrukcije varijabilne dužine. Procesori Pentium koriste instrukcije čija dužina ovisi o stvarnom načinu korištenja instrukcije. Kod takvih strojnih jezika, duljina je definirana op-kodom, koji kad se dekodira daje informaciju o broju byta koje iz memorije treba dohvatiti da bi se došlo do potpune instrukcije.