SPIE09 Obrada zvučnih signala
7. Razlike DSP i procesora
Ozren Bilan, viši predavač

Razlike DSP i *µprocesora*

DSP se izvodi primjenom matematičkih operacija, a obrada teksta i slični programi samo manipuliraju pohranjene podatke. To znači da računala za poslovne potrebe i opću primjenu nisu optimizirani za algoritme kao što je digitalno filtriranje i Fourierova analiza. Digitalni Signalni Procesori su posebna vrsta mikroprocesora specifično projektirani za obradu DSP zadaća. Doživjeli su ogroman razvoj posljednjih desetljeća nalazeći primjenu od mobitela do naprednih znanstvenih instrumenata.

Najuže grlo u izvršnim DSP algoritmima je **prijenos informacija iz memorije**. Zbog toga se klasična **Von Neumannova arhitektura** osobnih računala ne koristi pri DSP obradi. **Digitalni signalni procesor** (**DSP**) je specijalizirani mikroprocesor kojemu je interna arhitektura optimizirana za brze operacijske potrebe digitalne obrade signala. 2012. godine cijena im varira od 1 do 200 Eura (400 MHz/2400 MFLOPS). DSP su temeljeni na ***Harvard*** ili ***Super Harvard Arhitekturi***. Najvažnija područja poboljšanja su **instrukcijski *cache*** i **I/O kontroler**. Njihov učinkoviti prijenos podataka naziva se ***high memory-access bandwidth***.

Digitalni signalni procesor ili DSP je specijalizirani mikroprocesor s arhitekturom optimiziranom za brze operacije potrebne pri digitalnoj obradi signala. Obrađuje podatke u stvarnom vremenu pa je idealan za aplikacije koje ne trpe odgodu. Digitalni signalni procesori koriste digitalizirane video, glasovne, zvučne, temperaturne ili položajne signale i matematički ih obrađuju na najbrži mogući način. Cilj obrade je prikaz informacije ili pretvorba u drugačiji oblik signala.

Tipovi digitalnih signalnih procesora

Kategoriziraju se prema frekvenciji takta, iznosu RAM memorije, širinom podatkovne sabirnice, veličini ROM i Flash memorije, obliku čipa, MMAC/MIPS/FLOPS i I/Ι naponima. Tipične veličine RAM su 24 kB, 64 kB, 576 kB, 125 MB do 1 GB. Flash je u području 8 B to 1 GB, a tipične vrijednosti 4 kB. Digitalni signalni procesori su prikladni za programiranje.

Primjena digitalnih signalnih procesora

Najpoznatiji proizvođači	
Texas Instruments	46.7 %
AT&T	18.7 %
Motorola	15 %
Analog Devices	9.3 %
NEC	8.4%
ostali	1.9%

Digitalni signalni procesori projektiraju se za primjenu obrade zvučnih i video signala, sažimanje, obradu govora i prepoznavanje govora te digitalnu obradu slika, digitalne komunikacije, biomedicinu, seizmologiju, radar, sonar itd... Specifična primjena je prijenos govora u mobitelima, obrada seizmičkih podataka, analiza industrijskih procesa, primjena na slike u medicini CAT skeniranje, sažimanje zvučnih datoteka, računalna grafika.

Izbor optimalnog Digitalnog Signalnog Procesora prema namjeni

Pri izboru optimalnog DSP analiziraju se karakteristike frekvencije takta (100 Hz, 100 MHz, 150 MHz, 1000 MHz,...), iznos RAM memorije (1 kB, 24 kB, 576 kB,...), propusnost sabirnice *Data Bus Width* (od 8 b do 480 Mb), iznos ROM-a (96 B, 576 B, 96 kB,...) i ulazno izlaznog napona *I/O Voltage* (postoje DSP sve do 3.3 kV).

Primarne karakteristike *Digitalnih Signalnih Procesora*

brzina max	Sabirnica bita	U/I max napon	RAM	ROM	Flash (B)
100 Hz			1 kB	96 B	8 B
80 MHz	8 b	1.9 V	24 kB	576 B	4 kB
100 MHz	16 b	1.98 V	64 kB	2 kB	140 kB
120 MHz	24 b	2.5 V	576 kB	12 kB	1 GB
150 MHz	32 b	3 V	128 MB	96 kB	
250 MHz	64 ba>	3.3 V	1 GB	240 kB	
350 MHz	480 Mba>	3.6 V			
1000 MHz		3.3 kV			

Kako se DSP razlikuju od ostalih *µP*?

	MANIPULACIJA PODACIMA	MATEMATIČKI PRORAČUN
Tipična primjena	Obrada teksta, Baze podataka, Tablični kalkulatori Operacijski sustavi	DSP, dinamičko upravljanje, znanstvene simulacije
Glavna operacija	pomak podataka (*A←B*) ispitivanje (*Ako je A=B, onda je....*)	zbrajanje (*A + B = C*) množenje (*A x B = C*)

Manipuliranje podacima i matematički proračuni. Digitalna računala su korisna za dvije opće namjene: manipuliranje podacima i matematički proračuni. Manipuliranje podacima temeljeno je na pomaku podataka i ispitivanju nejednakosti, a matematički proračuni koriste množenje i zbrajanje.

Dosadašnji razvoj računala pokazao je da su digitalna računala ekstremno sposobna izvršavati zadaće na u dva područja:

- **manipulacija podataka**, kao što su tekst procesori i baze podataka i
- **matematički proračuni**, u znanosti, inženjeringu i DSP.

Svi mikroprocesori mogu obavljati obje zadaće; međutim, vrlo je teško i skupo napraviti uređaj koji je optimiziran za obje zadaće. Postoje tehnički kompromisi pri projektiranje sklopovlja, kao što su velićina instrukcija ili načina izvršenja prekida. Još značajniji su zahtjevi tržišta: cijena razvoja i izrade, kompetitivnost, trajnost proizvoda itd. U najširem smislu ti faktori su usmjerili tradicionalne **mikroprocesore**, u prvom redu na **manipulaciju podacima**. Slično tome, **DSP** se projektiraju kako bi izvršili potrebne **matematičke proračune** pri DSP.

Slika pokazuje najvažnije razlike dviju kategorija. Manipuliranje podacima uključuje pohranu i sortiranje informacija. Pogledajmo program za obradu teksta. Osnovna zadaća mu je pohrana informacija, organiziranje informacija (odreži i zalijepi, provjera pravopisa, izgled stranica, te dobava informacija (kao što su pohrana dokumenta u memoriju ili ispis na pisaču). **Zadaće se izvršavaju pomakom podataka s jednog mjesta na drugo i ispitivanjem nejednakosti (*A=B, A<B*, itd.)**.

Zamislimo sortiranje liste riječi abecednim redom. Svako slovo predstavlja 8 bitna znamenka, ASCII vrijednost prvog slova riječi. Poredak abecednim redom uključuje raspoređivanje riječi po kojemu ASCII vrijednosti jednoliko rastu od početka do kraja liste. To se radi ponavljanjem dva koraka sve dok se ne završi sortiranje po abecedi. Prvo se ispituju dva susjedna unosa da li su u abecednom redoslijedu (*IF A>B THEN* ...). Drugi korak, ako nisu, zamijeni ih tako da postanu (*A←→B*). Kada se ova dva koraka ponove dovoljno puta na svim zadanim susjednim riječima, lista će postati poredana po abecedi.

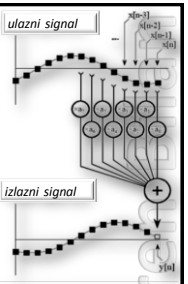
Pogledajmo kako se neki dokument ispisuje iz programa za obradu teksta. Računalo kontinuirano ispituje ulazni uređaj (miš ili tipkovnicu) za binarni kod koji znači "ispiši dokument." Ako ga detektira, program vrši pomak podataka iz memorije računala u pisač.

Pokazali smo samo dvije osnovne operacije: pomak podataka i ispitivanje nejednakosti. **Pri ovim primjenama matematika se koristi jako rijetko i ne djeluje na brzinu izvođenja programa**.

Usporedimo li to s brzinom izvođenja DSP algoritama, vidjet ćemo da su potpuno ograničeni brojem množenja i sumiranja potrebnih za izvršenje programa. Slika pokazuje primjenu FIR digitalnog filtra, najuobičajenije DSP tehnike. Ulazni signal je *x[]*, a izlaz je *y[]*. Zadaća je proračun uzorka na lokaciji *n* izlaznog signala, tj., *y[n]*. FIR filtar izvodi proračun množenjem odgovarajućih uzoraka ulaznog signala grupom koeficijenata, određenim: a_0, a_1, a_2, a_3.., i sumiranjem produkata. U obliku jednadžbe, *y[n]* se određuje:

$$y[n] = a_0x[n] + a_1x[n-1] + a_2x[n-2] + a_3x[n-3] + a_4x[n-4] + \cdots$$

Kažemo **ulazni signal se konvoluira s jezgrom filtra** (tj., impulsnim odzivom) koja se sastoji od: a_0, a_1, a_2, a_3,.. Prema aplikaciji u jezgri filtra može postojati nekoliko ili nekoliko tisuća koeficijenata. Iako u ovom algoritmu postoji i prijenos podataka, procjena nejednakosti, kao što su pohrane međurezultata i upravljačke petlje, matematičke operacije u potpunosti dominiraju vremenom izvođenja programa. Osim brzog izvođenja matematičke operacije, DSP mora imati predvidivo vrijeme izvođenja. Ako pokrenemo proces na računalu - pretvaranje tekst dokumenta iz jednog oblika u drugi, nije važno koliko će to potrajati 10 ms ili 10 *sekunda*; jednostavno **čekamo završetak procesa** prije nego što mu zadamo novi. Najveći broj DSP primjena **su obrade kontinuiranih signala, koje nemaju određeni početak i završetak** - kao projektiranje DSP audio sustava za pomoć nagluhima. Ako digitalni signal prima 20000 uzoraka/s, DSP mora biti sposoban održati izlaz 20000 uzoraka/s. Međutim, postoje razlozi zbog čega to ne bi smijelo biti brže. **Porastom brzine raste i cijena, potrošnja energije, poteškoće pri izradi i projektiranju, pa točno vrijeme brzine izvođenja odlučuje izbor uređaja i primjenjivi algoritam.**



FIR digitalni filtar. Pri FIR filtriranju, svaki uzorak izlaznog signala, y[n], određuje se množenjem uzoraka ulaznog signal, x[n], x[n-1], x[n-2], ..., s jezgrom filtra koeficijenti, a_0, a_1, a_2, a_3..., i sumiranjem produkata.

Kružni međuspremnik (*Circular Buffer*)

DSP projektiraju se kako bi brzo izveli FIR filtre i slične tehnike. Kako bi shvatili sklopovlje prvo moramo shvatiti algoritme. Napravit ćemo detaljni popis koraka koji su potrebni za primjenu FIR filtra, a nakon toga ćemo vidjeti kako se projektiraju DSP za najbrže moguće izvođenje tih koraka.

Potrebno je razlikovati *off-line* obradu od obrade u stvarnom vremenu. Pri *off-line* obradi **cjeloviti ulazni signal pohranjen je odjednom u računalu**. Tako geofizičar koristi seizmometar za snimanje pomaka zemlje za vrijeme potresa. Po završetku potresa, informacija se učitava u računalo i analizira. Drugi primjer *off-line* obrade su medicinske pretrage poput računalske tomografije ili magnetske rezonancije. Podaci se pribavljaju dok se snima pacijent, a rekonstruiraju se i analiziraju nakon toga. Pri tome je ključno što su onda **sve informacije simultano dostupne programu obrade**.

Pri obradi u stvarnom vremenu, **izlazni signal nastaje istovremeno dok se pribavlja ulazni signal** (*telefonska komunikacija, aparati za nagluhost, radar*). Za te primjene informacija mora biti trenutno dostupna iako može kasniti za vrlo kratak iznos. Tako se kašnjenje od 10 ms u telefonskom razgovoru ne može primijetiti. Slično tome, ne postoji razlika ako radarski signal kasni nekoliko sekunda prije nego se prikaže na ekranu operatoru. Primjene u realnom vremenu stalno učitaju uzorak, izvedu algoritam i daju uzorak na izlaz. Alternativno, **moguće je učitati grupu uzoraka, izvesti algoritam i daju na izlaz grupu uzoraka.**

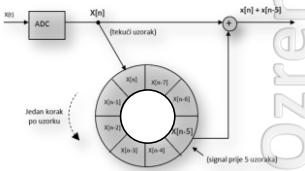
To je područje Digitalnih Signalnih Procesora.

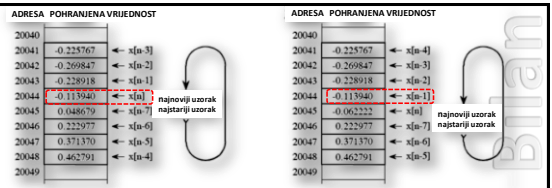
Jedan od najvažnijih postupaka pri digitalnoj obradi signala je proces linearne kombinacije tekućih (*trenutnih*) uzoraka s uzorcima signala koji kasne (*prošlih ili prethodnih uzoraka*). Postupak se matematički vrlo lako opisuje ali je mnogo korisnije za razumijevanje vizuelno predočiti princip rada tipičnog DSP procesora.

Temelj djelovanja je **kružni međuspremnik**, a u slučaju koji ćemo prikazati ima 8 memorijskih lokacija.

Iz AD pretvarača izlaze **uzorci signala**.

- Tekući uzorak *x[n]* pohranjuje se u **nultu lokaciju međuspremnika**.
- Kako dolaze novi uzorci pohranjena vrijednost pomiče se za jedan položaj u smjeru kazaljke u međuspremniku.
- Dakle, slijedeći položaj sadržava signal koji kasni za jedan korak ili jedan uzorak *x[n-1]*.





Zamislimo FIR filtar implementiran u realnom vremenu. Za proračun izlaznog uzorka potreban je pristup određenom broju prethodnih ulaznih uzoraka. Ako koristimo osam uzoraka u filtru: *a0, a1, … a7*, to znači da moramo znati vrijednost osam prethodnih uzoraka iz ulaznog signal, *x[n], x[n-1],…x[n-7]*. Te uzorke treba pohraniti u memoriji i stalno ažurirati kako pristižu novi uzorci. Najbolji način obrade pohranjenih uzorakaje **kružni međuspremnik**.

Slika ilustrira kružni međuspremnik za osam uzoraka. Smješten je u osam uzastopnih memorijskih lokacija, od 20041 do 20048. Slika (a) pokazuje kako je u nekom trenutku pohranjeno osam ulaznih uzoraka. Slika (b) pokazuje promjenu nakon što smo pribavili slijedeći uzorak. Ideja kružnog međuspremnika je u tome što je kraj linearne postave priključen na početak; memorijsku lokaciju 20041 možemo shvatiti kao prvu iduću nakon 20048, isto tako kao što je 20044 nakon 20045. Stanje memorijske postave pokazuje nam kazaljka (to je varijabla čija je vrijednost adresa) koja indicira gdje se nalazi najnoviji uzorak. Tako na slici (a) kazaljka sadržava adresu 20044, a na slici (b) 20045. Nakon što se pribavi novi uzorak, on zamjenjuje najstariji uzorak postave i kazaljka se pomiče za jedan broj unaprijed. Kružni međuspremnik je učinkovit jer nakon pribavljanja novog uzorka treba promijeniti samo jednu vrijednost.

Za upravljanje kružnim međuspremnikom potrebna su četiri parametra. **Prvo, mora postojati kazaljka (*pointer*) koja će pokazivati početak** kružnog međuspremnika u memoriji (u našem primjeru, 20041). **Drugo, mora postojati kazaljka koja će pokazivati završetak** kružnog međuspremnika (u našem primjeru, 20048) ili varijabla koja mu pohranjuje duljinu (npr. 8 bita). **Treće, mora se odrediti veličina koraka adresiranja**. Na slici 28-3 prikazan je jedinični korak. Tako adresa 20043 sadrži jedan uzorak, adresa 20044 sadržava idući itd… U praksi može nastati situacija da se adresiranje odnosi na bajtove, pa svaki uzorak može trebati dva do četiri bajta kako bi pohranio vrijednost uzorka. U takvim slučajevima veličina koraka adresiranja bit će dva ili četiri.

Navedene tri vrijednosti određuju veličinu i konfiguraciju kružnog međuspremnika i ne mijenjaju se za vrijeme rada programa. **Četvrta vrijednost, kazaljka posljednjeg uzorka mijenja se kako se pribavljaju novi uzorci**. Drugim riječima, **programska logika upravlja kako se ažurira četvrta vrijednost temeljem vrijednosti tri preostale**. Iako je logika vrlo jednostavna potrebna je ekstremna brzina. To je čitava bit priče;

DSP treba optimizirati kako bi upravljao kružnim međuspremnicima najvišom mogućom brzinom.

Kružni međuspremnici korisni su i za *off-line* obradu podataka. Pretpostavimo program u kojem su ulazni i izlazni signali potpuno pohranjeni u memoriji. Tada kružni međuspremnik nije potreban za proračun konvolucije jer je svaki uzorak uvijek dostupan iz memorije. Međutim, mnogi algoritmi primjenjuju se u stupnjevima pri kojima nastaju međusignali. Tako se rekurzivni filtri izvode kao serijski spoj *biquada*. Sirovi postupak je pohrana cjelovitih međuuzoraka u memoriji. Kružni međuspremnik omogućava drugačiji pristup jer pohranjuje samo međurezultate potrebne za proračun. Tako se smanjuje potrebna memorija po cijeni složenijeg algoritma. Važno je uočiti da su **kružni međuspremnici korisni za *off-line* obradu, a kritični su za primjenu u realnom vremenu**.

Pogledajmo **korake potrebne za primjenu FIR filtra korištenjem kružnog međuspremnika** za ulazni signal i koeficijente. Učinkovito izvršavanje individualnih zadaća je upravo ono što dijeli DSP od mikroprocesora. Za svaki novi uzorak potrebno je izvršiti slijedeće korake:

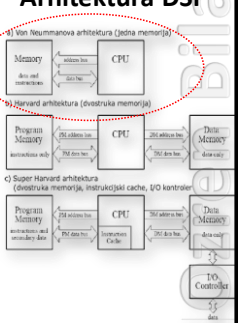
1. Pribaviti uzorak sa ADC; generirati prekid
2. Detektirati i omogućiti prekid
3. Prebaciti uzorak u kružni međuspremnik ulaznog signala
4. Ažurirati kazaljku međuspremnika ulaznog signala
5. Postaviti akumulator na nulu
6. **Upravljati petljom posredstvom svakog koeficijenta.**
7. **Dobaviti koeficijent kružnog međuspremnika koeficijenata**
8. **Ažurirati kazaljku koeficijenta kružnog međuspremnika ulaznog signala**
9. **Dobaviti uzorak iz kružnog međuspremnika ulaznog signala**
10. **Ažurirati kazaljku za kružni međuspremnik ulaznog kanala**
11. **Pomnožiti koeficijent uzorkom**
12. **Dodati produkt u akumulator**
13. Premjestiti izlazni uzorak (akumulator) u zadržavajući međuspremnik
14. Premjestiti izlazni uzorak iz međuspremnika u DAC

Nabrojeni koraci moraju se vrlo brzo izvoditi. Budući da će se korak 6-12 ponavljati mnogo puta (po jednom za svaki koeficijent filtra), mora se posvetiti pozornost ovim operacijama. Tradicionalni mikroprocesori općenito moraju izvršiti ovih 14 koraka u seriji jedan za drugim, dok su **DSP projektirani da ih izvode paralelno**. Ponekad se **sve operacije u petlji (koraci 6-12) obavljaju u jednom taktu**. To omogućava **interna arhitektura DSP**.

Von Neumannova Arhitektura DSP

Najuže grlo u izvršnim DSP algoritmima je **prijenos informacija u i iz memorije**. Pod pojmom podaci podrazumijevamo uzorke ulaznih signala, koeficijente filtra, programske instrukcije, binarne kodove koji ulaze u programski *sekvencer*… Za primjer pretpostavimo da je potrebno pomnožiti dva broja koja su pohranjena u memoriji. Da bi to napravili, moramo dobaviti tri binarne vrijednosti iz memorije, brojeve koji se množe i programske instrukcije koje pokazuju što treba napraviti.

Slika pokazuje kako se ova jednostavna zadaća izvršava u mikroprocesoru. Ovakva konfiguracija naziva se **Von Neumannova arhitektura**, po matematičaru *John Von Neumannu (1903-1957)*. **Von Neumann arhitektura** sastoji se od jednostruke memorije i jednostruke sabirnice koja služi za prijenos podataka iz CPU. **Množenje dva broja zahtijeva najmanje tri ciklusa takta**, za prijenos svakog broja preko sabirnice iz memorije u CPU. Pri tome ne ubrajamo vrijeme za prijenos rezultata u memoriju jer pretpostavljamo da on ostaje u CPU za dodatnu obradu, kao što je npr. suma produkata u FIR filtru. *Von Neumannova* arhitektura je zadovoljavajuća ako je korisnik zadovoljan serijskim izvođenjem svih zadaća.



Najveći broj današnjih računala ima *Von Neumannovu* arhitekturu. **Brže arhitekture su nam potrebne samo u slučajevima ekstremno brzih proračuna i ako smo voljni snositi troškove povećanja složenosti.**

Harvard Arhitektura DSP

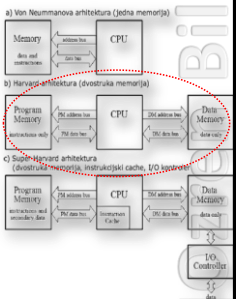
To nas vodi na **Harvard arhitekturu**, koju pokazuje slika (b). Dobila je ime prema istraživanju na *Harvard University* 1940. Kojeg je vodio *Howard Aiken* (1900-1973). Slika pokazuje kako je *Aiken* inzistirao na odvojenim memorijama za podatke i programske instrukcije, s posebnim sabirnicama.

Budući da **sabirnice djeluju nezavisno, programske instrukcije i podaci dobavljaju se istodobno, što poboljšava brzinu u odnosu na jednostruku sabirnicu.**

Suvremeni sustavi DSP koriste arhitekturu dvostruke sabirnice.

Problem temeljne Harvard arhitekture je taj što je

memorijska sabirnica podataka zaposlenija od programske sabirnica memorije



Super Harvard Arhitektura DSP

Slika pokazuje **Super Harvard Arhitektura**. Najvažnija područja poboljšanja su **instrukcijski *cache* i I/O kontroler**.

Problem Harvard arhitekture je taj što je **memorijska sabirnica podataka** zaposlenija od **programske sabirnica memorije**. Pri množenju dva broja, moraju se propustiti kroz sabirnicu podatkovne memorije (*data memory bus*) dvije binarne vrijednosti (brojevi koje se množe), a samo se jedna binarna vrijednost (programska instrukcija) propušta kroz sabirnicu programske memorije (*program memory bus*). Kako bi poboljšali situaciju, za početak ćemo prebaciti dio "podataka" u programsku memoriju. Npr., možemo prebaciti koeficijente filtra u programsku memoriju, a zadržati ulazni signal u podatkovnoj memoriji. (Na slici se ovi premještenti podaci nazivaju "sekundarni podaci").

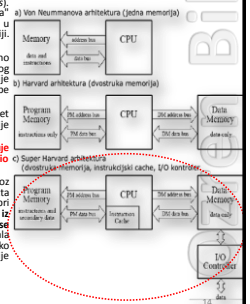
Na prvi pogled izgleda da se situacija ne popravlja jer sada moramo prenijeti jednu vrijednost preko podatkovne sabirnice (uzorak ulaznog signala), a dvije vrijednosti preko sabirnice programske memorije (programske instrukcije i koeficijent). Ako bi izvodili slučajne naredbe ne bi došlo ni do kakvog poboljšanja.

DSP algoritmi provode najviše vremena izvodeći petlje, pa će isti set programskih naredbi kontinuirano prolaziti od programske memorije prema procesoru CPU.

Super Harvard Arhitektura ima prednost tako što uključuje instrukcijski keš u CPU. To je mala memorija koja sadržava dio posljednjih programskih instrukcija.

Prvi put u petlji, programske naredbe moraju se propustiti kroz sabirnicu programske memorije. To usporava rad jer dolazi do konflikta budući da se koeficijenti također dobavljaju istim putem. Međutim, pri dodatnim izvođenjima petlje, **programske naredbe mogu se dobiti iz cache memorije naredbi**. To znači da se **cijeli sadržaj memorije koji se prijenosi u CPU može obaviti u jednom taktu**: uzorci ulaznog signala dolaze preko podatkovne sabirnice, koeficijenti dolaze preko programske sabirnice, a programske naredbe dolaze iz cache memorije naredbi.

Ovakav učinkoviti prijenos podataka naziva se ***high memory-access bandwidth.***



Cjelobrojni i realni brojevi (*Fixed i Real*)

DSP dijelimo u dvije kategorije prema načinu korištenja formata za pohranu i obradu brojeva u procesoru. To su cjelobrojni i realni brojevi. Fixed point DSP uobičajeno prikazuje svaki broj s minimalno 16 bitova, iako su moguće i druge dužine. Tako Motorola ima obitelj DSP koji koriste 24 bita. Četiri su uobičajena načina kojima se pomoću ovih 65536 mogućih kombinacija bitova može prikazati neki broj:

Cjelobrojna vrijednost bez predznaka (*unsigned integer*), omogućava pohranjenom broju popriminje bilo koje cjelobrojne vrijednosti od 0 do 65535.

Slično tome, **cjelobrojna vrijednost s predznakom** (*signed integer*) koristi komplement broja dva da u području uključi negativne brojeve od -32768 do 32767.

Pomoću **decimalnog zapisa bez predznaka** (*unsigned fraction*), 65536 razina jednoliko je raspodijeljeno između 0 i 1.

Konačno, **decimalni zapis s predznakom** (*signed fraction format*) dopušta negativne brojevi, jednoliko raspodijeljene između -1 i 1.

DSP s realnim brojevima koriste minimum 32 bita za pohranu svake vrijednosti. Tako je moguće prikazati za *fixed point*, 4294967296 brojeva. Ključno je što pri zapisu realni brojevi nisu jednoliko rašireni. U najčešćem formatu (*ANSI/IEEE Std. 754-1985*), najveći i najmanji brojevi su $\pm 3.4\times10^{38}$ i $\pm 1.2\times10^{-38}$. Prikazane vrijednosti raspodijeljene su nejednoliko između ekstrema, pa je razmak između bilo koja dva broja oko deset milijuna puta manji od tih brojeva. To je vrlo značajno, jer su veliki razmaci između velikih brojeva, a mali razmaci između malih brojeva.

Svi DSP koji rade s realnim brojevima mogu obrađivati i cjelobrojne vrijednosti, što je nužno za primjenu brojača, petlji i signala koji dolaze iz ADC i prosljeđuju se u DAC. Međutim, to ne znači da će se cjelobrojne matematičke operacije izvršiti brzo kao i operacije s realnim brojevima; što ovisi o unutrašnjoj arhitekturi.

Mnogi DSP su optimizirani za rad s realnim i cjelobrojnim vrijednostima te ih izvode podjednakom učinkovitošću. Ti procesori se nazivaju 32-bitni DSP, a ne procesori za realne brojeve.

DSP s cjelobrojnim vrijednostima su općenito jeftiniji , a oni s realnim brojevima imaju bolju preciznost, veće dinamičko područje i kraće vrijeme razvoja.

Aritmetika cjelobrojnim vrijednostima u računalima je mnogo brža nego s realnim brojevima ali brzina DSP je otprilike ista što je posljedica optimizacije sklopovlja za matematičke operacije. Unutrašnja arhitektura DSP za rad s realnim brojevima je mnogo složenija od onih za cjelobrojne vrijednosti. Svi registri i sabirnice moraju biti 32-bitni; množitelj i ALU moraju biti sposobni izvršavati brzu aritmetiku s realnim brojevima, a skup naredbi mora biti mnogo veći. Realni brojevi (32 bitni) imaju višu preciznost i mnogo više dinamičko područje. Programi s realnim brojevima često imaju kraće vrijeme razvoja, budući da programer ne treba voditi računa o prelijevu i greškama zaokruživanja.

32-bitni realni brojevi imaju mnogo viši odnos signal šum od 16-bitnih cjelobrojnih vrijednosti. Ako pohranimo broj u 32 bitnom formatu, razmak između ovog broja i susjednog je oko desetine milijuna vrijednosti broja. Da bi broj pohranili potrebno ga je zaokružiti na više ili na niže maksimalno za polovinu razmaka do susjednog. Drugim riječima, **svaki put kad neki broj pohranjujemo, dodajemo signalu šum**.

Isto se događa i kada broj pohranjujemo kao 16-bitnu fixed point vrijednost, osim što se dodaje mnogo više šuma. Razlog tome je mnogo veća širina razmaka među susjednim brojevima. Ako pohranjujemo broj 10000 u obliku cjelobrojne vrijednosti s predznakom (*signed integer*), koja se kreće od -32768 do 32767, razmak između brojeva bit će desettisućiti dio vrijednosti broja kojeg pohranjujemo. Pohranjujemo li broj 1000, razmak do susjednog broja bit će samo tisućiti dio vrijednosti. Šum u signalu uobičajeno prikazujemo standardnom devijacijom koja je u ovom slučaju kvantizacijskog šuma jednaka treći veličine razmaka. Proizlazi da će odnos signal/šum pri pohrani realnih brojeva biti odnos od 30 milijuna prema jedan, a za fixed point broj samo desettisuća prema jedan. Drugim riječima, **realni brojevi imaju 3000 puta manji šum kvantizacije**.

C ili Asembler

Programi u C su fleksibilniji i brže se pišu, a programi u asembleru imaju bolje karakteristike, mnogo su brži i koriste manje memorije pa su jeftiniji za primjenu.

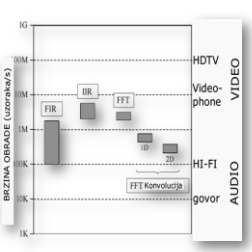
DSP se programiraju istim programskim jezicima kao i ostale znanstvene primjene, najčešće u asembleru ili C. Programi u asembleru su brži dok je programe u C-u lakše razviti i napisati. Pri tradicionalnim primjenama za osobna računala c je uvijek prvi izbor, a ako se koristi asembler obraničava se na kratke odsječke kod kojih je najvažnija brzina.

Međutim, DSP programi se razlikuju od tradicionalnih zadaća u dva važna aspekta. Prvo, programi su mnogo kraći, recimo stotinjak linija koda prema desecima tisuća. Drugo, brzina izvođenja je kritična. To je ujedno i razlog zašto se koristi DSP – velika brzina izvođenja. Zbog toga programeri prelaze na asembler.

Važna prednost korištenja viših programskih jezika poput C, Fortrana ili Basic su te što programer ne treba razumjeti arhitekturu mikroprocesora kojeg koristi; znanje o arhitekturi, prepuštena je *kompajleru*. Dakle, osoba koja piše program zna vrlo malo ili gotovo ništa o upravljanju memorijom, jer je zadaću preuzeo autor *kompajlera*. Programiranje u višem programskom jeziku zato je i mnogo lakše nego u asembleru jer je pola posla programera već obavila neka druga osoba. To je ujedno i razlog zbog kojeg je programiranje u višem programskom jeziku mnogo neučinkovitije jer programer ne zna kako će se program izvršavati. Možemo se upitati: koliko brzo će se izvršavati C programi u odnosu na asembler?

Odgovor je u najgorem slučaju **asembler je 2-3 puta brži**. Kako smo već naglasili, učinkovitost ovisi i o primjenjenom DSP. Pri tome su važni i programski alati, kao što su osobine *dibagera* koje nam pomažu pri razumijevanju kako se izvršavaju odsječci izvornog koda.

Koliko su brzi DSP?



Brzina DSP algoritma dobije se ***podijelimo li brzinu takta s potrebnim brojem ciklusa po uzorku****. Ilustracija pokazuje brzine četiri algoritma izvedena na DSP brzine 40 MHz.*

Uobičajeni način odgovora na pitanje su ***benchmark***, postupci za izražavanje brzine mikroprocesora u obliku jednog broja. Npr. brzina cjelobrojnih sustava izražava se u **MIPS** (*milijuna cjelobrojnih operacija/s*). Slično tome, brzina realnih sustava izražava se u **MFLOPS** (*milijun operacija s realnim brojevima/s*).

Ideja *benchmarka* je omogućiti usporedbu između mikroprocesora da bi vidjeli koji je najbolji. Različiti mikroprocesori pokazuju najbolje karakteristike u različitim kategorijama. Bez iskustva korištenja *benchmarka* doći ćemo do pogrešnih zaključaka. Bolji pristup ovom problemu bio bi **kolika je brzina izvođenja algoritma kojeg želimo izvoditi**. Ako primjena DSP zahtjeva FIR filter, pogledajmo koliki je točan broj ciklusa takta potreban procesoru za izvođenje zadaće

Brzina mikroprocesora podvostručuje se svake tri godine. To znači da je potrebno obratiti posebnu pažnju na postupak koji koristimo u ovom primjeru. U praksi se stvarni brojevi uvijek mijenjaju pa treba ponoviti proračun za svaki novi projekt. Neki DSP mogu izvesti *multiply-accumulate* operacije u samo jednom ciklusu takta. Budući da najveći broj FIR filtra koristi 25 do 400 koeficijenta, potrebno je 25 do 400 ciklusa takta za svaki obrađeni uzorak. Da bi dobili izlaz filtera moguće je podijeliti takt (npr. 40 MHz ili više) s brojem ciklusa takta koji su potrebni za svaki uzorak. To će nam dati maksimalni FIR protok od 100k do 1.6M uzoraka/s. Te FIR izlazne vrijednosti prikazane su na slici.

Tipični IIR filtri koriste 5 do 17 koeficijenta. Budući da su petlje relativno kratke, dodat ćemo 3 ciklusa po uzorku. Tako dobivamo 8 do 20 ciklusa takta potrebnih po uzorku obrađenih podataka. Pri 40 MHz taktu maksimalni IIR propust je od 1.8M do 3.1M uzoraka/s. Vrijednosti pokazuje slika. Pomnožite ove vrijednosti s 25 za danas tipični 1GHz takt procesora.

Proizvodnja Digital Signal Procesora

DSP tržište je vrlo veliko i stalno raste. Najveća primjena su mobiteli, multimedijalna računala, reprodukcija glazbe visoke vjernosti CD, DVD, SACD, kućno kino te digitalna televizija. Manje profitabilna područja obuhvaćaju znanstvenu instrumentaciju i uređaje za akviziciju podataka.

DSP se može nabaviti u tri oblika: kao jezgra, procesor i proizvod za pločicu. Izraz jezgra podrazumijeva dio procesora u kojem se izvode ključne zadaće, što uključuje registre podataka, množila, ALU, generator adresa i programski sekvencer. Cjeloviti procesor zahtjeva kombiniranje jezgre s memorijom i sučelje prema okolini. Iako se jezgra i dijelovi periferije projektiraju odvojeno, izrađuju se u istom čipu, pa je procesor u jednom integriranom sklopu.

Ako projekt treba DSP, najvjerojatnije će se nabaviti DSP u obliku procesora – koji sadržava jezgru, memoriju i sučelje. Česti su u pakovanju *240 lead Metric PQFP* dimenzija 35×35×4 mm. Najčešći način primjene DSP je ubacivanje DSP čipa u tiskanu pločicu uređaja, za sklop koji ste projektirali. Proizvođači integriranih sklopova ne žele cijeli procesor, nego samo jezgre koje ugrađuju u svoje čipove. Takve jezgre napravljene po narudžbi također su česte.

Razlike između DSP i mikroprocesora nisu uvijek čvrste. Slijedećim riječima Intel je opisao uvođenje MMX tehnologije u Pentium procesore: "… ***have added 57 powerful new instructions specifically designed to manipulate i process video, audio i graphical data efficiently. These instructions are oriented to highly parallel, repetitive sequences often found in multimedia operations.***„

Danas vidimo sve više i više klasičnih DSP funkcija u klasičnim mikroprocesorima i mikrokontrolerima. Internet i multimedijska primjena zahtijevaju takve promjene.