Arhitekture i algoritmi
DSP 2

Auditorne vežbe AU-10[0]
Tehnike optimizacije u ASM

Odsek za računarsku tehniku i računarske komunikacije

Optimizacije

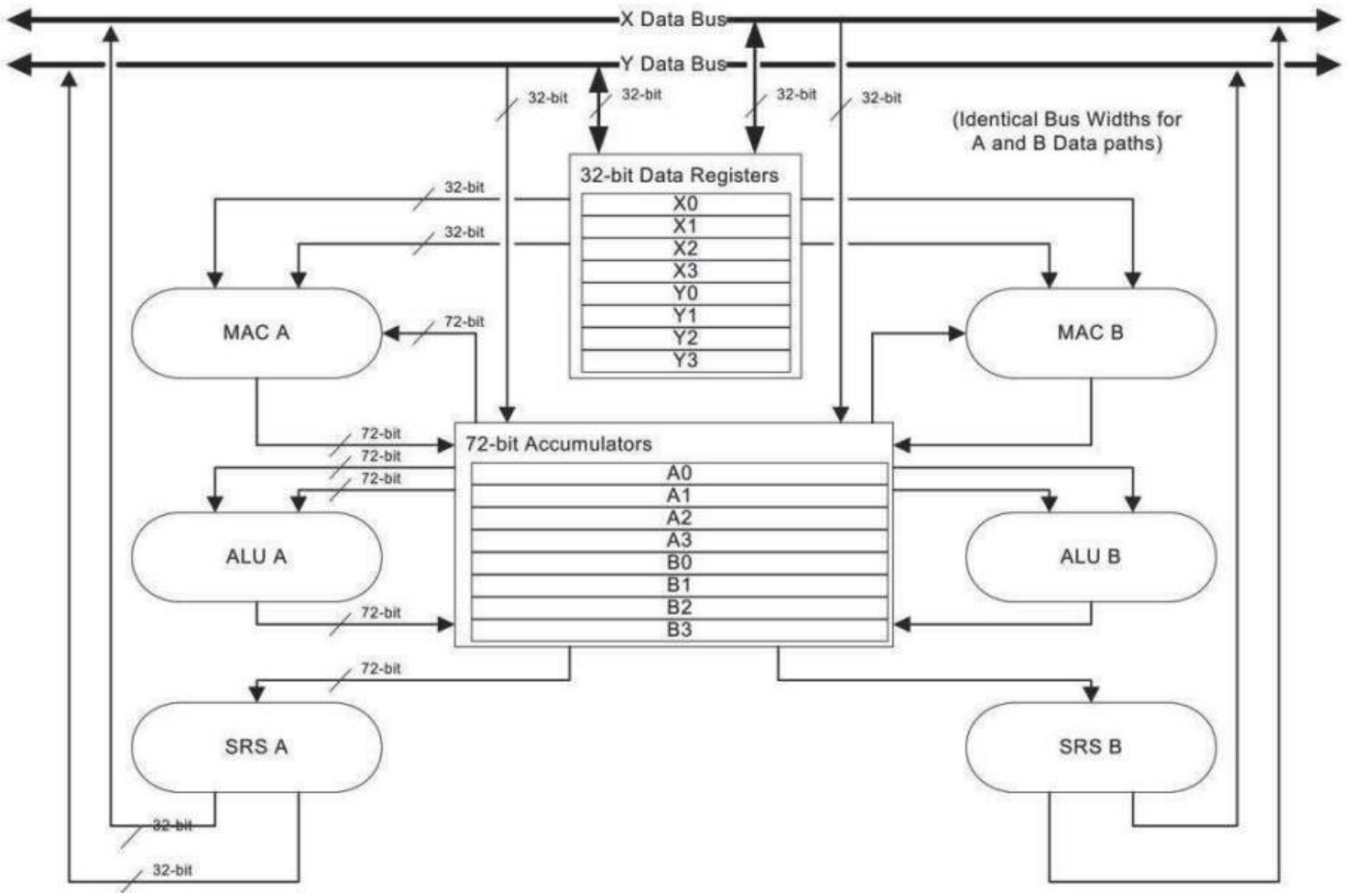
- Cilj je iskoristiti pipeline DSP jezgra što je moguće više
- Osmotriti operacije unutar funkcije obrade i obratiti pažnju na redosled propagacija vrednosti i rezultata
- Obratiti pažnju na invarijantne delove petlje!
- Uvek gledati kako memorijske prelaze uskladiti i učešljati da se izvršavaju u jednoj instrukciji

Crystal DSP pipeline

- 2 MAC operacije + 2 memorijska prelaza + 2 ažuriranja pokazivača

```
a0+=y0*x1;  b0+=y0*x0;  x0=xmem[i0]; i0-=1;  y0=ymem[i4]; i4+=1
```

1. Operacije koje zauzimaju po 8 gornjih bita instrukcione reči
   a. Paralelni višenamenski prenosi. Dozvoljavaju po jedan prenos sa X i jedan prenos sa memorijskom zonom Y u jednoj instrukciji. U istoj instrukciji može biti izvršeno i ažuriranje odgovarajućih indeksnih registara, a važi i to da mogu biti smeštene u paraleli sa MAC/ALU instrukcijom. Restrikcije vezane za protok podataka iz/u memorijsku zonu X ili Y su sledeće:
   b. Memorijska zona X može biti adresirana samo korišćenjem indeksnih registara i0 i i1
   c. Memorijska zona Y može biti adresirana samo korišćenjem indeksnih registara i4 i i5
   d. Memorijski prenos ka memorijskoj zoni X može biti obavljen jedino korišćenjem A akumulatora
   e. Memorijski prenos ka memorijskoj zoni Y može biti obavljen jedino korišćenjem B akumulatora
   f. Akumulatori (a0-a3, b0-b3) mogu učestvovati jedino kao izvorišni resursi.
   g. Registri za podatke (x0-x3, y0-y3) mogu učestvovati



Slika 2.4 - Blok dijagram DSP jezgra CS48x

Pre-fill

- Priprema podataka pre petlje (pre-fill)

```
 do (i4),>block_loop
                                                          i4=i6
                                                          x0=ymem[i2]; i2+=n
    a1 =+ x0;                                             x1=ymem[i2]; i2+=n
    b1 =+ x1;                                             y0=ymem[i4]; i4+=1
    a0=y0*x0;    b0=y0*x1;    x1=xmem[i0]; i0-=1;         y0=ymem[i4]; i4+=1
    do (i1),>
      a0+=y0*x1;  b0+=y0*x0;  x0=xmem[i0]; i0-=1;   y0=ymem[i4]; i4+=1
%:    a0+=y0*x0;  b0+=y0*x1;  x1=xmem[i0]; i0-=1;   y0=ymem[i4]; i4+=1
    a0+=y0*x1;  b0+=y0*x0;                                i0+=n
    b1 =+ a0;    a0 = b1
                                xmem[i0]=a1; i0+=1; ymem[i5]=b1; i5+=n
%block_loop:                    xmem[i0]=a0;        ymem[i5]=b0; i5+=n
                                                          xmem[i3]=i0
```