Lab 6 (w/emubox) – digital signal processors

Resources

Motorola 56k instruction set

<table>
<tr><th>registers</th><th colspan="3">symbols</th></tr>
<tr><td>X0, X1, Y0, Y1
A, B</td><td>x, y
a, b</td><td>s</td><td rowspan="2">g, h</td></tr>
<tr><td>R0, ..., R7
N0, ..., N7</td><td>r
n</td><td>i</td></tr>
</table>

abs, asl, asr, clr, neg, rnd: `abs a ;`
add, sub: `add s, a ;`
mpy, mpyr, mac, macr: `mpy ±x,y,a ;`
nop

```
move x:ea,g ; from memory
move g,x:ea ; to memory
move ea ; (update Rn)
move g,h ;
move #c,g ;
```

ea – Effective Address (see table)⟶

assembly		meaning		mode
ea	X&Y	ea	R update	
(r)-n			r=r-n ;	
(r)+n	(yes)		r=r+n ;	
(r)-	(yes)		r=r-1 ;	
(r)+	(yes)		r=r+1 ;	
(r)	(yes)	r		indirect
(r+n)		r+n		indexed
c		c		absolute

```
macr -x0,x0,a   a,x:(r3)-   y:(r5)+n5,x0
```

Example `c5_fir7.asm` - FIR order 7

Please note: first column is for labels, the rest of line must be after a tab.

```
N       equ     8
        org     x:0
samples ds      N
        org     y:0
coeffs  dc      0.0286,0.0716,0.1683,0.2458,0.2458,0.1683,0.0716,0.0286
        org     p:$100
        init
        move    #samples,r0
        move    #coeffs,r4
        move    #N-1,m0
        move    m0,m4
        .repeat
          in    a
          move  a,x:(r0)
          clr   a               x:(r0)+,x0      y:(r4)+,y0
          .loop #N-1
            mac x0,y0,a         x:(r0)+,x0      y:(r4)+,y0
          .endl
          macr  x0,y0,a         (r0)-
          nop ; DSP56321  pipelining need this !
          out   a
        forever
```

Example `c5_iir3.asm` **- IIR order 3**

```
N       equ     3
        org     x:0
states  ds      N
        org     y:0
coeffs  dc      0.8739,0.9217,0.2671,-0.2036,0.2036,-0.1868,0.1868
        org     p:$100
        init
        move    #states,r0
        move    #coeffs,r4
        move    #N-1,m0
        move    #2*N,m4
        .repeat
          in    a
          move                x:(r0)+,x0      y:(r4)+,y0
          .loop #N-1
            mac -x0,y0,a      x:(r0)+,x0      y:(r4)+,y0
          .endl
          macr  -x0,y0,a      x:(r0)+,x0      y:(r4)+,y0
          nop ; DSP56321!
          clr   a             a,y1
          .loop #N-1
            mac +x0,y0,a      x:(r0)+,x0      y:(r4)+,y0
          .endl
          mac   +x0,y0,a      x:(r0)-,x0      y:(r4)+,y0
          macr  +y1,y0,a      y1,x:(r0)
          nop ; DSP56321 !
          out   a
        forever
```

How to obtain a working box

- make your program (please follow the directions, some tools need it **just this** way)
  - Use `windows explorer` to create "New⟶ Text file" in your working directory (you can e.g. create a "New folder" on the desktop)
  - Change the name of the file like *project*`.ASM`, agree with the warning
  - **Drag** your file to the SciTe icon – it opens the SciTe editor to edit your file
  - translate *project*`.ASM` into *project*`.CLD`– use SciTe menu "Narzedzia" (Tools) ⟶ "Buduj" (build)
  - view *project*`.LST` – check for errors (open it e.g. with SciTe)
  - on errors, iterate through edit-translate-check
- simulate program run
  - prepare data (Matlab, running it from "Narzedzia" (Tools) menu of SciTe makes `X.DAT` file in proper directory)
    `save56(cos(0.1*(0:99)));` if you need a test signal
    or `save56(delta56);` if you want to find impulse response
    then `quit`

- execute simulator ("Symulator" in SciTe menu); the simulator is set up so that each "in" instruction will read from `X.DAT`, each "out" will write to `Y.DAT`
  - choose "overwrite" (if asked) – to replace old `Y.DAT`
  - `step 10000 cy` - means "10000 clock cycles"
  - `quit`
- view output data (Matlab again)
  `load56;` loads `X.DAT` and `Y.DAT`, then makes graphs (simple or FFT – if `X.DAT` was a delta)

- Now run the program on the real signal processor
  - Use "Uruchom" in SciTe menu to load *project*`.CLD` into EMU BOX
  - check if it works - with a real signal
    - connect signal source to "In1"
    - connect "Out1" to the oscilloscope
    - remember that A/D and D/A introduce 0.7 ms delay
  - verify the frequency response with a network analyzer (implemented with a PC) ⟶ only if your program is a linear filter
    - connect output from computer D/A to the "In1" of EMU BOX (test signal is generated by computer)
    - connect "Out1" to the computer A/D input (filter response is measured by computer
    - run `Anator` with "Device"⟶ "Network analyzer" set from the menu
    - don't worry if there are errors above 10-12 kHz :-).

Experiments

1. translate, simulate and execute a simple program for division by 2

```
; remember about tab!
        org     p:$100 ;program start address (lower are reserved)
        init ;   init codec
        repeat
        in a ; read a sample into a
        asr a ;  arithmetic shift right
        nop ;
        out a ;
        forever
```

   Test with sinusoid:

   - `sin(1:100)` or like this in Matlab,
   - generator of approx. 2 kHz sine in hardware; change frequency and amplitude, note interesting results

   Sketch results.

2. Change program to:

   - multiply signal by 2

- rectify signal

Test with sinusoid, sketch selected (nontrivial) results.

3. understand, translate, simulate and execute a simple FIR filter program (`C5_FIR7.ASM`)
   - try to understand program design; ask teacher if something is not clear
   - check amplitude characteristics of simulated filter (make sketch); here we plot the FFT of a recorded impulse response – what important assumption do we make?
   - use generator and oscilloscope to verify the characteristics in 2–4 selected frequency points (mark on the previous sketch, comment)
   - use network analyzer to measure characteristics (sketch or comment versus the previous sketch)
   - sketch filter structure graph
   - make a sketch/sketches showing usage of data buffer `samples`
4. understand, translate, simulate and execute a simple IIR filter program (`C5_IIR3.ASM`)
   - check amplitude characteristics of simulated filter
   - use generator and oscilloscope to verify the characteristics in 2–4 frequency points
   - use network analyzer to measure characteristics
   - try to understand program design; what purpose does the buffer `states` serve?
   - sketch filter structure graph
   - use Matlab to plot filter characteristics (theoretical – from coefficients), compare with the simulated and measured ones
5. Design a coefficient set for different characteristics and verify it with real filter

File: lab6emubox LaTeXed on January 5, 2010