Procesory Sygnałowe w aplikacjach przemysłowych

Wprowadzenie

IET
Katedra Elektroniki
Kraków 2015
dr inż. Roman Rumian
Digital Signal Processing – the processing or manipulation of signals using digital techniques
• Selecting a DSP – several choices:
  – Fixed-point;
  – Floating point;
  – Application-specific devices
    (e.g. FFT processors, speech recognizers, etc.).

• Main DSP Manufacturers:
  – Texas Instruments (http://www.ti.com)
  – Motorola (http://www.freescale.com)
  – Analog Devices (http://www.analog.com)
Typical DSP Operations

- Filtering
- Energy of Signal
- Frequency transforms

\[ y(n) = \sum_{i=0}^{L-1} a_i x(n - i) \]

**Pseudo C code**

```c
for (n=0; n<N; n++) /*block filtering*/
{
    s=0;
    for (i=0; i<L; i++)
    {
        s += a[i] * x[n-i];
    }
    y[n] = s;
}
```
Traditional DSP Architecture

\[ y(n) = \sum_{i} a_{i} x(n-i) \]
## SHARC Benchmarks

<table>
<thead>
<tr>
<th></th>
<th>ADSP-21160N SIMD</th>
<th>ADSP-21161N SIMD</th>
<th>ADSP-21262 SIMD</th>
<th>ADSP-21266 SIMD</th>
<th>ADSP-21371 SIMD</th>
<th>ADSP-21375 SIMD</th>
<th>ADSP-21364 SIMD</th>
<th>ADSP-21365 SIMD</th>
<th>ADSP-21368 SIMD</th>
<th>ADSP-21369 SIMD</th>
<th>ADSP-2146x SIMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Cycle</td>
<td>100 MHz</td>
<td>150 MHz</td>
<td>200 MHz</td>
<td>266 MHz</td>
<td>333 MHz</td>
<td>400 MHz</td>
<td>450 MHz</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instruction Cycle Time</td>
<td>10 ns</td>
<td>6.67 ns</td>
<td>5 ns</td>
<td>3.75 ns</td>
<td>3 ns</td>
<td>2.5 ns</td>
<td>2.22 ns</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MFLOPS Sustained</td>
<td>400 MFLOPS</td>
<td>600 MFLOPS</td>
<td>800 MFLOPS</td>
<td>1064 MFLOPS</td>
<td>1332 MFLOPS</td>
<td>1600 MFLOPS</td>
<td>1800 MFLOPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MFLOPS Peak</td>
<td>600 MFLOPS</td>
<td>900 MFLOPS</td>
<td>1200 MFLOPS</td>
<td>1596 MFLOPS</td>
<td>1998 MFLOPS</td>
<td>2400 MFLOPS</td>
<td>2700 MFLOPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1024 Point Complex FFT (Radix 4, with bit reversal)</td>
<td>92 µs</td>
<td>61.3 µs</td>
<td>46 µs</td>
<td>34.5 µs</td>
<td>28 µs</td>
<td>23 µs</td>
<td>20.44 µs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIR Filter (per tap)</td>
<td>5 ns</td>
<td>3.3 ns</td>
<td>2.5 ns</td>
<td>1.88 ns</td>
<td>1.5 ns</td>
<td>1.25 ns</td>
<td>1.11 ns</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IIR Filter (per biquad)</td>
<td>20 ns</td>
<td>13.3 ns</td>
<td>10 ns</td>
<td>7.5 ns</td>
<td>6 ns</td>
<td>5 ns</td>
<td>4.43 ns</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Matrix Multiply (pipelined) [3x3] * [3x1]</td>
<td>45 ns</td>
<td>30 ns</td>
<td>22.5 ns</td>
<td>16.91 ns</td>
<td>13.5 ns</td>
<td>11.25 ns</td>
<td>10.00 ns</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Divide (y/x)</td>
<td>30 ns</td>
<td>20 ns</td>
<td>15 ns</td>
<td>11.27 ns</td>
<td>9 ns</td>
<td>7.5 ns</td>
<td>6.67 ns</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inverse Square Root</td>
<td>45 ns</td>
<td>30 ns</td>
<td>22.5 ns</td>
<td>16.91 ns</td>
<td>13.5 ns</td>
<td>11.25 ns</td>
<td>10.00 ns</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SHARC
'S'uper 'H'arvard 'ARC'hitecture
Register File and COMPUTE Units

- Key issues:
  - 5 data paths FROM COMPUTE units
  - 5 data paths TO COMPUTE units
  - Highly parallel operations UNDER THE RIGHT CONDITIONS
Register File – BIT STORAGE

- **Key issues**
  - 40 bits wide
  - Top 32 bits used for integer
  - Top 32 bits used for float
  - 40 bits for precision float
  - 32 registers available
    - 16 at a time

- **A Register is always 40 bits**
  - can be processed as a float
  - can be processed as an integer
  - Must convert integer<-> float
Instruction format

- Instructions are 48 bits wide
- 23 bits – COMPUTE field – are available for computer operations – See appendix B-1
- Single-function format
  22 21-20 19-12 11-8 7-4 3-0
  
  11-8 destination  7-4, 3-0 – source register
  19-12 opcode associated with computation unit (bits 21-20) (ALU, MAC or SHIFTER)
  Bit 22 always a 0 for single function format
Algebraic notation terminated by semicolon:

\[ R1 = DM(M0, I0), \quad R2 = PM(M8, I8);!\text{comment} \]

\texttt{label:} \quad R3 = R1 + R2;

- DAG1 and DAG2 registers
- Data memory access
- Program memory access
# Sample ALU Instructions

## ALU INSTRUCTIONS – COMPUTE OPERATION

### INTEGER

<table>
<thead>
<tr>
<th>ALU Instruction</th>
<th>ALU Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rn = Rx + Ry</td>
<td>Rn = Rx + Ry</td>
</tr>
<tr>
<td>Rx = Rx – Ry</td>
<td>Rx = Rx – Ry</td>
</tr>
<tr>
<td>Rn = Rx + Ry + CI (Carry In)</td>
<td>Rn = Rx + Ry + CI (Carry In)</td>
</tr>
<tr>
<td>Rn = Rx - Ry + CI - 1</td>
<td>Rn = Rx - Ry + CI - 1</td>
</tr>
<tr>
<td>Rn = (Rx + Ry) / 2</td>
<td>Rn = (Rx + Ry) / 2</td>
</tr>
<tr>
<td>COMP(Rx, Ry)</td>
<td>COMP(Rx, Ry)</td>
</tr>
<tr>
<td>Rn = Rx + CI – 1</td>
<td>Rn = Rx + CI – 1</td>
</tr>
<tr>
<td>Rn = Rx + 1</td>
<td>Rn = Rx + 1</td>
</tr>
<tr>
<td>Rn = Rx – 1</td>
<td>Rn = Rx – 1</td>
</tr>
<tr>
<td>Rn = -Rx</td>
<td>Rn = -Rx</td>
</tr>
<tr>
<td>Rn = ABS Rx</td>
<td>Rn = ABS Rx</td>
</tr>
<tr>
<td>Rn = PASS Rx</td>
<td>Rn = PASS Rx</td>
</tr>
<tr>
<td>Rn = Rx AND Ry</td>
<td>Rn = Rx AND Ry</td>
</tr>
<tr>
<td>Rn = Rx OR Ry</td>
<td>Rn = Rx OR Ry</td>
</tr>
<tr>
<td>Rn = NOT Rx</td>
<td>Rn = NOT Rx</td>
</tr>
<tr>
<td>Rn = MIN(Rx, Ry)</td>
<td>Rn = MIN(Rx, Ry)</td>
</tr>
<tr>
<td>Rn = MAX(Rx, Ry)</td>
<td>Rn = MAX(Rx, Ry)</td>
</tr>
<tr>
<td>Rn = CLIP Rx by Ry</td>
<td>Rn = CLIP Rx by Ry</td>
</tr>
</tbody>
</table>

### FLOAT

<table>
<thead>
<tr>
<th>ALU Instruction</th>
<th>ALU Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fn = Fx + Fy</td>
<td>Fn = Fx + Fy</td>
</tr>
<tr>
<td>Fn = Fx - Fy</td>
<td>Fn = Fx - Fy</td>
</tr>
<tr>
<td>Fn = ABS(Fx + Fy)</td>
<td>Fn = ABS(Fx + Fy)</td>
</tr>
<tr>
<td>Fn = ABS(Fx – Fy)</td>
<td>Fn = ABS(Fx – Fy)</td>
</tr>
<tr>
<td>Fn = (Fx + Fy) / 2</td>
<td>Fn = (Fx + Fy) / 2</td>
</tr>
<tr>
<td>COMP(Fx, Fy)</td>
<td>COMP(Fx, Fy)</td>
</tr>
<tr>
<td>Fn = - Fx</td>
<td>Fn = - Fx</td>
</tr>
<tr>
<td>Fn = ABS Fx</td>
<td>Fn = ABS Fx</td>
</tr>
<tr>
<td>Fn = PASS Fx</td>
<td>Fn = PASS Fx</td>
</tr>
<tr>
<td>Fn = RND Fx</td>
<td>Fn = RND Fx</td>
</tr>
<tr>
<td>Fn = SCALB Fx BY Ry</td>
<td>Fn = SCALB Fx BY Ry</td>
</tr>
<tr>
<td>Rn = MANT Fx</td>
<td>Rn = MANT Fx</td>
</tr>
<tr>
<td>Rn = LOGB Fx</td>
<td>Rn = LOGB Fx</td>
</tr>
<tr>
<td>Rn = FIX Fx BY Ry</td>
<td>Rn = FIX Fx BY Ry</td>
</tr>
<tr>
<td>Fn = FLOAT Rx BY Ry</td>
<td>Fn = FLOAT Rx BY Ry</td>
</tr>
<tr>
<td>Rn = TRUNC Fx</td>
<td>Rn = TRUNC Fx</td>
</tr>
<tr>
<td>Fn = RECIPS Fx</td>
<td>Fn = RECIPS Fx</td>
</tr>
<tr>
<td>Fn = RSQRRTS Fx</td>
<td>Fn = RSQRRTS Fx</td>
</tr>
<tr>
<td>Fn = Fx COPYSIGN Fy</td>
<td>Fn = Fx COPYSIGN Fy</td>
</tr>
<tr>
<td>Fn = MIN(Fx, Fy)</td>
<td>Fn = MIN(Fx, Fy)</td>
</tr>
<tr>
<td>Fn = MAX(Fx, Fy)</td>
<td>Fn = MAX(Fx, Fy)</td>
</tr>
<tr>
<td>Fn = CLIP Fx by Fy</td>
<td>Fn = CLIP Fx by Fy</td>
</tr>
</tbody>
</table>
MULTIFUNCTION - COMPUTE OPERATION

On certain registers only, unlike standard COMPUTE
Multiplication \( FN = FQ \times FR \), with \( FQ=\{0,1,2,3\} \) and \( FR=\{4,5,6,7\} \)
ALU Compute \( FN = FX \ op FY \), \( FX=\{8,9,10,11\}, FY=\{12,13,14,15\} \)

Dual Add/Subtract – integer or float allowed
\( FN = FX + FY \), \( FM = FX - FY \);

Parallel Multiplier / ALU operation – integer of float allowed
\( FN = FQ \times FR \), \( FM = \) Any ALU operation using \( FX \) and \( FY \)

Parallel Multiplier with Dual Add/ Subtract
\( FN = FQ \times FR \), \( FM = FX + FY \), \( FO = FX - FY \);

IMMEDIATE MOVE – NOT VALID WITH “IF COMPUTE”
\[ \text{ureg} \leftrightarrow \text{dm(\langle addr32\rangle)}; \quad \text{ureg} \leftrightarrow \text{pm(\langle addr24\rangle)}; \]
\[ \text{dm(\langle addr32\rangle, Ia)\leftrightarrow \text{ureg}}; \quad \text{pm(\langle addr24\rangle, Ic)\leftrightarrow \text{ureg}}; \]
\[ \text{ureg} = \text{\langle data32\rangle} \]
MAC INSTRUCTIONS – INTEGER COMPUTE

Rn = Rx * Ry
MRB = Rx * Ry
Rn = MRB + Rx * Ry
MRB = MRB + Rx * Ry
Rn = Rn

MRF = Rn
MRB = Rn
Rn = SAT MRF
MRB = SAT MRB
Rn = RND MRB
MRB = RND MRB
Rn = MR

MAC INSTRUCTIONS -- FLOAT COMPUTE

Fn = Fx * Fy
SHIFTER OPERATIONS

Rn = LSHIFT Rx BY Ry/<data8>
Rn = Rn OR LSHIFT Rx BY Ry/<data8>
Rn = ASHIFT Rx BY Ry/<data8>
Rn = ROT Rx BY Ry/<data8>
Rn = BCLR Rx BY Ry/<data8>
Rn = BSET Rx BY Ry/<data8>
Rn = BTGL Rx BY Rx/<data8>
Rn = BCLR Rx BY Ry/<bit6>:<len6> (SE)
Rn = BSET Rx BY Ry/<bit6>:<len6> (SE)
Rn = EXP Rx (EX)
Rn = LEFT0 Rx
Fn = UNPACK Rx
MISCELLANEOUS

BIT SET/CLR/TGL/TST,XOR srg <data32>;

<data32> = MASK for SET/CLR/TGL/TST

MODIFY (Ia, <data32>)/(Ic, <data24>)

BITREV (Ia, <data32>)/(Ic, <data24>)

PUSH/POP LOOP, PUSH/POP STS PUSH/POP PCSTK,

FLUSH CACHE;

NOP; IDLE; IDLE16 – idle till interrupt – low power

SHARC NUMBER FORMATS

STANDARD SHARC REGISTERS ARE 40-bits wide

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

INTEGER

s bexp (8-bits)

FLOAT

frac (23-bits)
DATA – R0 to R15
INDEX (Address) – I0 to I7, I8 to I15
MODIFY -- M0 to M7 M8 to M15
LENGTH – L0 to L7, L8 to L15
BASE – B0 to B7, B8 to B15  (Setting Bx also set Ix)
Ia/Mb refers to DAG1 (dm) registers – Ic/Md refers to DAG2 (pm)

PROGRAM SEQUENCER – PC, PCSTK, PCSTKP, FADDR, DADDR, LADDR, CURLCNTR, LCNTR
BUS EXCHANGE -- PX1, PX2, PX
TIMER – TPERIOD, TCOUNT
SYSTEM REGISTERS – sreg -- MODE1, MODE2, IRPTL, IMASK, IMASKP, ASTAT, STKY, USTAT1, USTAT2
PRE AND POST MODIFY OPERATIONS

PRE-MOD -- (Mb, Ia) – Use address (Mb + Ia) – Leave Ia unchanged

IF La register = 0 – causes normal array operation

POST-MOD -- (Ia, Mb) – Use address (Ia) – Change Ia to Ia + Mb

IF La register != 0 – causes circular buffer operations

POST-MOD – (Ia, Mb) – Use address (Ia) – Change Ia to Ia + Mb
then perform Ia – La or Ia + La until Ia in range Ba to Ba + La – 1
Memory Accesses

Under the right conditions -- 3 memory accesses at same time
Program Memory, Data Memory, Instruction Cache
PLUS up to 2 ALU + 1 MAC operations at the same time
PLUS background DMA activity
There is only 1 memory, but it is broken into 2 sections
DAG1 -- best for accessing Data memory section (0 -- 7)
DAG2 -- best for accessing Program memory section (8 -- 15)
MUST be used in this fashion for simultaneous memory ops
Also an alternate set of DAGs (SHADOW DAGs)
Register and Register Ops in DAG1

SPECIAL CIRC BUFFER STUFF

SPECIAL FFT BIT
DAG register info

- Index registers
  - I0 -- I7 (dm -- data mem), I8 -- I15 (pm -- program mem)
  - “like” 68K address registers A0 -- A6
- Modify registers M0 -- M7, M8 -- M15
  - Can be offset registers (c.f 68K (4, SP))
  - Can be used for high speed post increment
- Special Hardware for Circular Buffers
  - Base registers B0 -- B7, B8 -- B15
  - Length registers L0 -- L7, L8 -- L15
  - See labs 2 -- 4 and associated lectures
Address Versus Word Size

The processor’s internal memory accommodates the following word sizes:

- 64-bit long word data (LW)
- 40-bit extended-precision normal word data (NW, 48-bit)
- 32-bit normal word data (NW, 32-bit)
- 16-bit short word data (SW, 16-bit)

Only the address space determines which memory word size is accessed. An important item to note is that the DAG automatically adjusts the output address per the word size of the address location (short word, normal word, or long word). This address adjustment allows internal memory to use the address directly as shown in the following example.

```plaintext
I15=LW_addr;
pm(i15,0)=r0; /* 64-bit transfer */

I7=NW_addr;
dm(i7,0)=r8; /* 32-bit transfer */

I7=SW_addr;
dm(i7,0)=r14; /* 16-bit transfer */
```
## Internal Memory Map

Table 4. Internal Memory Space (5 MBits—ADSP-21486/ADSP-21487/ADSP-21489)\(^1\)

<table>
<thead>
<tr>
<th>Long Word (64 Bits)</th>
<th>Extended Precision Normal or Instruction Word (48 Bits)</th>
<th>Normal Word (32 Bits)</th>
<th>Short Word (16 Bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block 0 ROM (Reserved)</td>
<td>Block 0 ROM (Reserved)</td>
<td>Block 0 ROM (Reserved)</td>
<td>Block 0 ROM (Reserved)</td>
</tr>
<tr>
<td>0x0000 0000–0x0004 7FFF</td>
<td>0x0000 0000–0x0008 AAA9</td>
<td>0x0008 0000–0x0008 FFFF</td>
<td>0x0010 0000–0x0011 FFFF</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0004 8000–0x0004 8FFF</td>
<td>0x0008 AAAA–0x0008 BFFF</td>
<td>0x0009 0000–0x0009 1FFF</td>
<td>0x0012 0000–0x0012 3FFF</td>
</tr>
<tr>
<td>Block 0 SRAM</td>
<td>Block 0 SRAM</td>
<td>Block 0 SRAM</td>
<td>Block 0 SRAM</td>
</tr>
<tr>
<td>0x0004 9000–0x0004 EFFF</td>
<td>0x0008 C000–0x0009 3FFF</td>
<td>0x0009 2000–0x0009 DFFF</td>
<td>0x0012 4000–0x0013 BFFF</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0004 F000–0x0004 FFFF</td>
<td>0x0009 4000–0x0009 FFFF</td>
<td>0x0009 E000–0x0009 FFFF</td>
<td>0x0013 C000–0x0013 FFFF</td>
</tr>
<tr>
<td>Block 1 ROM (Reserved)</td>
<td>Block 1 ROM (Reserved)</td>
<td>Block 1 ROM (Reserved)</td>
<td>Block 1 ROM (Reserved)</td>
</tr>
<tr>
<td>0x0005 0000–0x0005 7FFF</td>
<td>0x000A 0000–0x000A AAA9</td>
<td>0x000A 0000–0x000A FFFF</td>
<td>0x0014 0000–0x0015 FFFF</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0005 8000–0x0005 8FFF</td>
<td>0x000A AAAA–0x000A BFFF</td>
<td>0x000B 0000–0x000B 1FFF</td>
<td>0x0016 0000–0x0016 3FFF</td>
</tr>
<tr>
<td>Block 1 SRAM</td>
<td>Block 1 SRAM</td>
<td>Block 1 SRAM</td>
<td>Block 1 SRAM</td>
</tr>
<tr>
<td>0x0005 9000–0x0005 EFFF</td>
<td>0x000A C000–0x000B 3FFF</td>
<td>0x000B 2000–0x000B DFFF</td>
<td>0x0016 4000–0x0017 BFFF</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0005 F000–0x0005 FFFF</td>
<td>0x000B 4000–0x000B FFFF</td>
<td>0x000B E000–0x000B FFFF</td>
<td>0x0017 C000–0x0017 FFFF</td>
</tr>
<tr>
<td>Block 2 SRAM</td>
<td>Block 2 SRAM</td>
<td>Block 2 SRAM</td>
<td>Block 2 SRAM</td>
</tr>
<tr>
<td>0x0006 0000–0x0006 3FFF</td>
<td>0x000C 0000–0x000C 5554</td>
<td>0x000C 0000–0x000C 7FFF</td>
<td>0x0018 0000–0x0018 FFFF</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0006 4000–0x0006 FFFF</td>
<td>0x000C 5555–0x000D FFFF</td>
<td>0x000C 8000–0x000D FFFF</td>
<td>0x0019 0000–0x001B FFFF</td>
</tr>
<tr>
<td>Block 3 SRAM</td>
<td>Block 3 SRAM</td>
<td>Block 3 SRAM</td>
<td>Block 3 SRAM</td>
</tr>
<tr>
<td>0x0007 0000–0x0007 3FFF</td>
<td>0x000E 0000–0x000E 5554</td>
<td>0x000E 0000–0x000E 7FFF</td>
<td>0x001C 0000–0x001C FFFF</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0007 4000–0x0007 FFFF</td>
<td>0x000E 5555–0x000F FFFF</td>
<td>0x000E 8000–0x000F FFFF</td>
<td>0x001D 0000–0x001F FFFF</td>
</tr>
</tbody>
</table>
INSTRUCTIONS AND DELAY JUMP SLOT

R2 = 1;
R8 = pass R2;
If NE jump(pc, _LABEL) (DB);
    R8 = 2; Execute whether jump or not
    R7 = 1; Execute whether jump or not
R8 = 3;
_LABEL:
Warning: R7 = 1, whether jump OR NOT,
R8 = 3 if jump DOES NOT OCCUR,
R8 = 2 if jump occurs and not 1
COMPUTE AND MOVE INSTRUCTIONS (PARALLEL)

```
compute, dm(Ia, Mb) ↔ dreg1, pm(Ic, Md) ↔ dreg2;

IF condition compute;  N.B. italics = optional part of instruction

N.B. “IF” operation affects the WHOLE instruction

IF condition compute,   dm(Pre/Post with MREGISTERs)↔ ureg;
IF condition compute,   pm(Pre/Post with MREGISTERs)↔ ureg;

N.B. ureg can’t be from same DAG as Pre/Post registers

IF condition compute,   dm(Pre/Post with <data6>) ↔ ureg;
IF condition compute,   pm(Pre/Post with <data6>) ↔ ureg;

IF condition compute,   dreg↔ dm(IREG, MREGISTER)
IF condition compute,   dreg↔ pm(IREG, MREGISTER)
IF condition compute,   reg1 = ureg2;
IF condition shiftimm,   dm(IREG, MREGISTER)↔ dreg;
IF condition shiftimm,   pm(IREG, MREGISTER)↔ dreg;
IF condition compute,   MODIFY (IREG, MREGISTER);
```
N.B. italics = optional part of instruction

N.B. “IF” operation affects the WHOLE instruction

IF condition JUMP <addr24> (DB/LA/CI);
IF condition JUMP (PC, <reladdr24>) (DB/LA/CI);
IF condition CALL <addr24> (DB);
IF condition CALL (PC, <reladdr24>) (DB);
IF condition JUMP (Md, Ic) (DB/LA/CI), compute;
IF condition JUMP (PC, <reladdr24>) (DB/LA/CI) ELSE compute;
IF condition CALL (Md, Ic) (DB/LA/CI), compute;
IF condition CALL (PC, <reladdr24>) (DB/LA/CI) ELSE compute;
IF condition JUMP (Md. Ic), ELSE compute, dm(Ia, Mb) = dreg;
IF condition JUMP (Md. Ic), ELSE compute, dreg = dm(Ia, Ib);
IF condition RTS (DB/LR), compute;
IF condition RTS (DB/LR), ELSE compute;
IF condition RTI (DB/LR), compute;
IF condition RTI (DB/LR), ELSE compute;

LCNTR = <data16>, DO <addr24> UNTIL LCE;
LCNTR = ureg, DO <PC, <reladdr24> UNTIL LCE;

General Form of DO

DO <addr24> UNTIL termination;

General Form of DO

DO (PC, <reladdr24>) until termination

(DB) Delayed branch
(LA) Loop abort (pop loop and PC stacks on branch)
(CI) Clear interrupt
Instrukcje skoków bezwarunkowych i warunkowych

JUMP etykieta //skok bezwarunkowy

CALL etykieta //bezwarunkowe wywołanie podprogramu

IF NE JUMP etykieta //skok warunkowy

IF AC CALL etykieta //warunkowe wywołanie podprogramu

Niektóre pozostałe instrukcje także mogą być wykonywane warunkowo, np. :

IF EQ DM(I0,M0) = R2;
IF EQ R8 = R2;
# ADSP-SC58x & ADSP-2158x Family Comparison

<table>
<thead>
<tr>
<th>Device</th>
<th>ADSP-SC589</th>
<th>ADSP-SC587</th>
<th>ADSP-SC584</th>
<th>ADSP-SC583</th>
<th>ADSP-SC582</th>
<th>ADSP-21587</th>
<th>ADSP-21584</th>
<th>ADSP-21583</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM® Cortex®-A5 (64 kB L1, 256 kB L2)</td>
<td>450 MHz</td>
<td>450 MHz</td>
<td>450 MHz</td>
<td>300 MHz</td>
<td>450 MHz</td>
<td>300 MHz</td>
<td>450 MHz</td>
<td>—</td>
</tr>
<tr>
<td>SHARC+ Processors</td>
<td>2x 450 MHz</td>
<td>2x 450 MHz</td>
<td>2x 450 MHz</td>
<td>2x 300 MHz</td>
<td>2x 450 MHz</td>
<td>2x 300 MHz</td>
<td>1x 450 MHz</td>
<td>2x 450 MHz</td>
</tr>
<tr>
<td>L1 SRAM/Cache (with parity)</td>
<td>2x 640 kB</td>
<td>2x 640 kB</td>
<td>2x 640 kB</td>
<td>2x 384 kB</td>
<td>640 kB</td>
<td>2x 640 kB</td>
<td>2x 640 kB</td>
<td>2x 384 kB</td>
</tr>
<tr>
<td>L2 Shared SRAM (with ECC)</td>
<td>256 kB</td>
<td>256 kB</td>
<td>256 kB</td>
<td>256 kB</td>
<td>256 kB</td>
<td>256 kB</td>
<td>256 kB</td>
<td>256 kB</td>
</tr>
<tr>
<td>L2 Shared ROM</td>
<td>512 kB</td>
<td>512 kB</td>
<td>512 kB</td>
<td>512 kB</td>
<td>512 kB</td>
<td>512 kB</td>
<td>512 kB</td>
<td>512 kB</td>
</tr>
<tr>
<td>L3 16-Bit Ports DDR3/DDR2/LPDDR1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>GigE AVB Ethernet (MAC)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>10/100 Ethernet (MAC)</td>
<td>1</td>
<td>1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>USB 2.0 HS and PHY</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>SDIO/eMMC</td>
<td>1</td>
<td>1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>MLB (Auto Only)</td>
<td>—</td>
<td>6 p/3 p</td>
<td>6 p/3 p</td>
<td>6 p/3 p</td>
<td>6 p/3 p</td>
<td>—</td>
<td>—</td>
<td>6 p/3 p</td>
</tr>
<tr>
<td>PCIe</td>
<td>1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>GPIO</td>
<td>102</td>
<td>102</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>102</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>Common Peripherals</td>
<td>2x digital audio interfaces (each with 4x SPORT/PS, S/PDIF, 2x ASRC, 2x PCG), 3x PC, quad SPI, 2x dual SPI, 2x CAN 2.0, 3x UART, 2x link ports, ePPI, 3x ePWM, 2x WDT, 8x timer, 1x counter, RTC, ACM, 8-channel, 12-bit ADC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware Accelerators</td>
<td>High performance FFT/IFFT, FIR/IIR filtering, harmonic analysis engine, sinc filter, security crypto engines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grade (Comm/Indust/Auto)</td>
<td>C/I</td>
<td>C/I/A</td>
<td>C/I/A</td>
<td>C/I/A</td>
<td>C/I</td>
<td>C/I</td>
<td>C/I/A</td>
<td>C/I/A</td>
</tr>
<tr>
<td>Package (19 mm x 19 mm, 0.8 p)</td>
<td>529 BGA</td>
<td>529 BGA</td>
<td>349 BGA</td>
<td>349 BGA</td>
<td>349 BGA</td>
<td>529 BGA</td>
<td>349 BGA</td>
<td>349 BGA</td>
</tr>
</tbody>
</table>