RT VIRTUAL LABS
(DIGITAL SIGNAL PROCESSING)
VC33 BOARD

Nagesh S Nayak
Department of Electrical Engineering,
IIT Bombay

OUTLINE

- TMS320VC33
  - Features
  - Architecture
  - Addressing Modes
  - Programming Details
  - Sample Codes

TMS320VC33 FEATURES

- 32-bit floating point processor (32-bit instruction word)
- 24-bit address bus (16MB addressable space)
- 1 x Serial Port, 2 x 32 bit timers, DMA
- 75 MHz clock (External crystal 15MHz internally x 5 using PLL)
- 13ns Instruction Cycle
- Multiple operations per instruction (e.g. Multiply, MAC)
- 1 instruction executed per Instruction cycle

ARCHITECTURE - REGISTERS

- R0-R7 – GP Extended Precision registers
- AR0-AR7 – Auxiliary registers (Address generation)
- DP – Data Page Pointer (Memory organized as 256 pages of length 64k each)
- IR0-IR1 – Index registers (Address offset)
- SP – Stack Pointer (PUSH pre-increment POP post-decrement)
- RC – Repeat Counter (Repeat a block of code given by addresses in 2 registers....CX reg. for 8086)
- ST – Status Register
- IF – Interrupt Flag Register

ADDRESSING MODES

- Register (3 operand)

  absi R0

- Direct

  Address Generation - Direct address- DP (8 LSB shifted) and offset (16 bit) given in instruction

  Advantage: allowing you to access a large address space without requiring a change of the page pointer

  ldp @cos;

  ldf @cos, R0

ADDRESSING MODES

- Indirect

  Use of ARs and IRs (Lower 24 bits of AR)

  Advantage: Modify auxiliary registers in parallel with operations within the main CPU

  stf R0,*+AR1(1);

  stf R0,*AR1++(1);

  stf R0,*++AR0(IR0);

- Immediate

  16 bit---------- addi 1,R0;

  24 bit---------- call 0x810000;

- PC relative addressing

  Branch bz label1;

- Parallel instructions (Loading, Arithmetic, Arithmetic/Logical + Store)
  - AND src2,src1,dst1 || STI src3,dst2

TMS320VC33

- Programming Details

  asm File: Assembly language file, Assembler directives

  map file: addresses to be used during programming (RAM Block 1 reserved for monitor program)

- Code 1: Multiply two floating point numbers
- Code 2: Convolution

PROGRAM CODES

```
* Floating point multiply program
            .def MAIN                       ; Makes MAIN global
            .data
*************Variables initialization***********
x           .float      1.2e-3
y           .float      3.5e-2
z           .float      0
            .text
MAIN:
            ldp         @x; Load Data Page Pointer
            ldf         @x, r0; X-> R0
            mpyf        @y, r0; X*Y->R0
            stf         r0, @z; R0->Z
STOP:       br          STOP;
            .end
```

MAP FILE

```
ENTRY(MAIN)

SECTIONS
{
        . = 0x800000;
        .data : { *(data) }
        . = 0x801000;
        .bss : { *(bss) }
        . = 0x809800;
        .text : { *(.text) }
}
```

CONVOLUTION CODE

```
        .def MAIN               ; Makes MAIN global

********************** DATA **********************
        .data
x       .float  0.1,0.5,0.1,0.2
x_adr   .word x                 ;coefficients
m       .int    4; number of entries in x
h       .float  0.1,0.5,0.2,0.25
h_adr   .word h                 ;input samples
l       .int    4; number of entries in h
        .bss    y,8             ; Using BSS Space for result
y_adr   .word   y
```

```
********************* TEXT ************************
********************* MAIN ************************
        .text
MAIN:
**********Initializing output locations to 00h************
        LDI     8,R4            ;
        LDI     @y_adr,AR3      ;
        LDF     00h,R2;
init:   STF     R2,*AR3++(1);
        SUBI    01h,R4          ;
        CMPI    00h,R4;
        BNZ     init;
*****************************************************
```

```
            LDI     00h,R2;             counter for x
restart:    LDI     R2,IR0;
            LDI     x_adr,AR1;
            LDI     y_adr,AR0;
            LDI     *++AR1(IR0),R3; Dummy Write
            LDI     *++AR0(IR0),R3; Dummy Write
            LDI     h_adr,AR2;
            LDI     00h,R1;             counter for h
finish_h:   MPYF    *AR2++(1),*AR1++(0),R0;
            ADDF    *AR0++(0),R0;
            STF     R0,*AR0++(1);
            ADDI    01h,R1;
            CMPI    @I,R1;              Check if loop for h complete
```

```
BNZ     finish_h;
ADDI    01h,R2;Change offset to point to next element in x & y
CMPI    @m,R2;          Check if loop for x complete
BNZ     restart;
STOP:  br       STOP            ;ends the program
.end
```

VIRTUAL LABS

Thank You

REFERENCES

- TMS320VC33 User Guide