**Programming Model**

**General Purpose Register Array**

DSD1 has an array of 32, 64 bit general purpose integer registers, coupled with an array of 32, 64 bit floating point registers.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Usage |  |  |  |
| r0 | always zero |  | fp0 |  |
| r1 | return value |  | fp1 | return value |
| r2 |  |  | fp2 |  |
| r3 | temporary register |  | fp3 | temporary register |
| r4 | temporary register |  | fp4 | temporary register |
| r5 | temporary register |  | fp5 | temporary register |
| r6 | temporary register |  | fp6 | temporary register |
| r7 | temporary register |  | fp7 | temporary register |
| r8 | temporary register |  | fp8 | temporary register |
| r9 | temporary register |  | fp9 | temporary register |
| r10 | temporary register |  | fp10 | temporary register |
| r11 | register var |  | fp11 |  |
| r12 | register var |  | fp12 |  |
| r13 | register var |  | fp13 |  |
| r14 | register var |  | fp14 |  |
| r15 | register var |  | fp15 |  |
| r16 | register var |  | fp16 |  |
| r17 | register var |  | fp17 |  |
| r18 | register var |  | fp18 |  |
| r19 |  |  | fp19 |  |
| r20 |  |  | fp20 |  |
| r21 |  |  | fp21 |  |
| r22 |  |  | fp22 |  |
| r23 |  |  | fp23 |  |
| r24 | task register (TR)1 |  | fp24 |  |
| r25 | thread pointer (TP) |  | fp25 |  |
| r26 | global pointer (GP) |  | fp26 |  |
| r27 | frame pointer (BP) |  | fp27 |  |
| r28 |  |  | fp28 |  |
| r29 |  |  | fp29 |  |
| r30 | stack pointer (SP) |  | fp30 |  |
| r31 |  |  | fp31 |  |

1 Not updateable in user mode.

## Next Address Register Array

The branch register array stores instruction addresses. Most of the branch registers have pre-set uses and may be automatically updated while the processor is running.

|  |  |  |
| --- | --- | --- |
|  | Usage | Auto Updated |
| b0 | always zero |  |
| b1 |  |  |
| b2 | catch link address |  |
| b3 | debug return address | Y |
| b4 | interrupt return address | Y |
| b5 | exception return address | Y |
| b6 | return address | Y |
| b7 | program counter | Y |

## Condition Code Register Array

There are eight four bit condition code registers (cc0 to cc7). ALU instructions update the condition code register associated with the general purpose target register. For the compare instructions the condition code register to update may be explicitly identified. Conditional branch instructions test and branch based on the value in a condition code register.

|  |  |
| --- | --- |
|  | Associated Target Register |
| cc0 | r0,r8,r16,r24 |
| cc1 | r1,r9,r17,r25 |
| cc2 | r2,r10,r18,r26 |
| cc3 | r3,r11,r19,r27 |
| cc4 | r4,r12,r20,r28 |
| cc5 | r5,r13,r21,r29 |
| cc6 | r6,r14,r22,r30 |
| cc7 | r7,r15,r23,r31 |

### Condition Code Register Format (Integer):

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 2 | 1 | 0 |
| N | V | Z | C |

## Segment Selector Registers

The ISA supports six ten bit segment selector registers as shown in the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SPR | SEG | 31 10 | 9 0 |  |
| 18 | 0 |  | DS10 | data selector |
| 19 | 1 |  | ES10 |  |
| 1A | 2 |  | FS10 |  |
| 1B | 3 |  | GS10 |  |
| 1C | 4 |  | --- | unsupported |
| 1D | 5 |  | --- | unsupported |
| 1E | 6 |  | SS10 | stack selector |
| 1F | 7 |  | CS10 | code selector |

The default segment selector register used for address formation is selected by base register association for load / store instructions. For flow control instructions the code segment is always used. For instance if the instruction uses the stack pointer as a base, the processor will automatically select the stack selector as the segment register for the instruction. This default setup may be overridden using a selector prefix in assembler code.

# Memory Management

The ISA provides for a segmented memory management model. Memory is divided into a number of segments. The segments are described in the segment descriptor table. A 10 bit selector value is used to index into the descriptor table in order to retrieve the information on the segments.

## The Descriptor Table

The descriptor table contains information on the location and size for segments. This table is a special 1k word dual-ported memory embedded within the processor. Entries in the table have the following format:

|  |  |  |
| --- | --- | --- |
|  | 63 32 | 31 0 |
| w0 | Limit32 | Base32 |
| w1 | Limit32 | Base32 |
| … |  |  |

The descriptor table is located at a fixed address of $0000\_1000.

10 bit selectors are used to index into the table in order to determine the characteristics of the segment.

# Instruction Formats

Instructions vary in length and are a multiple of a byte (eight bits) in size.

## Branch Instruction Formats

There are JMP and CALL instructions which branch within +/-2^28 bytes of the program counter. JMP and CALL instructions may be preceded by a selector prefix. There is a JMP or CALL register format which allows an absolute JMP or CALL to a value in a branch register. JMP or CALL register may also be used to return from a subroutine.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | |  |  |  |
|  | Offset32 | | | | | | Opcode8 | CALL |
|  | Offset32 | | | | | | Opcode8 | JMP |
|  | | Offset24 | | | | | Opcode8 | CALL |
|  | | Offset24 | | | | | Opcode8 | JMP |
|  | | | Offset16 | | | | Opcode8 | CALL |
|  | | | Offset16 | | | | Opcode8 | JMP |
|  | | | | ~ | Bn3 | Bd3 | Opcode8 | CALLR |
|  | | | | ~ | Bn3 | 03 | Opcode8 | JMPR |
|  | | | | ~ | 63 | 03 | Opcode8 | RET |
|  | | | | ~ | 53 | 03 | Opcode8 | ERET |
|  | | | | ~ | 43 | 03 | Opcode8 | IRET |
|  | | | Disp13 | | | CC3 | Opcode8 | Bcc |
|  | | | | Disp8 | | | Opcode8 | Bcc cc3, |
|  | | Selector24 | | | | | Opcode8 | JSP |

## Compare Instructions

Compare instruction place the result status into a condition code register.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  | | |  |  |  |  |
|  | | | | | CC3 | Ra5 | Opcode8 | TST |
|  | | Immed8 | | | CC3 | Ra5 | Opcode8 | CMP |
|  | Immed16 | | | | CC3 | Ra5 | Opcode8 | CMP |
|  | | ~ | CC3 | Rb5 | | Ra5 | Opcode8 | CMP |
|  | |  | | |  |  |  |  |

Arithmetic / Logical

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | Immed6 | Rt5 | Ra5 | Opcode8 |  |
|  | Immed14 | | Rt5 | Ra5 | Opcode8 |  |
|  | |  |  |  |  |  |

The processor is a slow one by today’s standards.

Instructions vary in lengths that are a multiple of a byte in size. A handful of instructions like the Debug trap, or segment prefixes are a single byte in length. Most commonly performed branches are two bytes in length. Other instructions vary in length as needed.

There are 32, 64 bit general purpose registers.

The processor has an x86 style segmentation model. Selectors are 24 bit rather than 16 bits to allow a larger number of segments and privilege levels.

Calls and interrupts always store the same information on the stack. A 32 bit processor state field and 24 bit code segment selector are stored in the first word on the stack, followed by a 64 bit offset address in the next stack word. Only a single return instruction is required. The return instruction checks the state information on the stack to determine the type of return (interrupt, debug, exception, or regular).

The processor uses multiple condition code registers to record result status of operations. The branch instructions specify which condition code register to evaluate. There is a shortened form of branch instructions that evaluate condition code register #3 implicitly.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | x0 | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | x9 | xA | xB | xC | xD | xE | xF |
| 0x | BRK |  |  | DBG | ADD #6 | SUB #6 | CMP #8 | MUL #6 | DIV #6 | MOD #6 |  |  | AND #6 | OR #6 | XOR #6 |  |
| 1x |  |  |  |  | ADD #14 | SUB #14 | CMP #16 | MUL #14 | DIV #14 | MOD #14 |  |  | AND #14 | OR #14 | XOR #14 |  |
| 2x |  |  |  |  | ADD | SUB | CMP | MUL | DIV | MOD |  |  | AND | OR | XOR |  |
| 3x |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4x | TST |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5x | CALL32 | CALL24 | CALL16 | CALLR | JMP32 | JMP24 | JMP16 |  | RET |  |  |  |  |  |  |  |
| 6x | BRA |  | BHI | BLS | BHS | BLO | BNE | BEQ | BVC | BVS | BPL | BMI | BGE | BLT | BGT | BLE |
| 7x | BRA |  | BHI3 | BLS3 | BHS3 | BLO3 | BNE3 | BEQ3 | BVC3 | BVS3 | BPL3 | BMI3 | BGE3 | BLT3 | BGT3 | BLE3 |
| 8x | LB | LBU | LC | LCU | LH | LHU | LW |  | LBX | LBUX | LCX | LCUX | LHX | LHUX | LWX | LWARX |
| 9x | SB | SC | SH | SW |  |  |  |  | SBX | SCX | SHX | SWX | SWCRX |  |  |  |
| Ax |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Bx |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Cx |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Dx |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Ex |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fx | CS | DS | EF | FS | GS | SS |  |  | IMM8 | IMM16 | IMM24 | IMM32 | IMM40 | IMM48 | IMM56 | NOP |