Title: SDP - Samarth Designed Processor

Author: Samarth Javagal

Date: 2025-05-24

***1. Abstract***

This paper outlines the architecture and design of the SDP series. These CPUs and

GPUs are created to be educational, and provide deep concepts in terms of computer

architecture

***2. Introduction***

The SDP series (Samarth Designed Processor) includes Central Processing Units (SDP-xC), Graphical Processing Units (SDP-xG), and Custom LED Displays (SDP-xD). There are 3 versions of each in the SDP. The SDP-1C, SDP-1G, SDP-1D, SDP-2C, SDP-2G, SDP-2D, SDP-3C, SDP-3G, and the SDP-3D. The corresponding versions of the CPUs, GPUs, and Displays are compatible with each other

***3. SDP-1C, SDP-1G, and SDP-1D***

***3.1 SDP-1C***

The SDP-1C has:

* A bit width of 8
* 8 general purpose registers (A, B, C, D, E, F, G, and H)
* An ALU
* The MAR
* The input MDR (IDR)
* The output MDR (ODR)
* A program counter of width 2 bytes (PC)
* A stack separate from the main RAM with depth 256 (SK)
* A stack pointer consisting of 8 bits (SP)
* An instruction register (IR)
* Control logic
* IO Ports

***3.1.1 General Purpose Registers***

The 8 general purpose registers are used with the Register Selector (RS). The RS has an input of 4 bits, and correspondingly selects the register. See Table-1 for the mappings. The RS only connects the selected register to the bus and enables operations for the register, The only exceptions are the A and B Registers, which are always connected to the bus.

***Table-1:***

| Register | Input For The Register Selector |
| --- | --- |
| A | 0x0 |
| B | 0x1 |
| C | 0x2 |
| D | 0x3 |
| E | 0x4 |
| F | 0x5 |
| G | 0x6 |
| H | 0x7 |

***3.1.2 The ALU***

The inputs of the ALU are the A and B registers. See Table-2 for the operations performed by the ALU, and the registers it uses. The ALU has 3 flags, the zero flag, carry flag, and sign flag

***Table-2:***

| Operation | Registers Used | Description |
| --- | --- | --- |
| Addition | A, B | Adds the contents of the A and B registers, and stores the result in the A register |
| Subtraction | A, B | Subtracts the contents of the A and B registers, and stores the result in the A register |
| Rotate Right | A | Rotates the A register right once and stores the result in the A register |
| Rotate Left | A | Rotates the A register left once and stores the result in the A register |
| XOR | A, B | XORs the contents of the A register and B register and stores the result in the A register |
| NAND | A, B | NANDs the contents of the A and B registers, and stores the result in the A register |
| Shift Right | A | Shifts the A register right once and stores the result in the A register |
| Shift Left | A | Shift the A register left once and stores the result in the A register |

***3.1.3 The MAR***

The MAR of the SDP-1C is 16 bits. And is only allowed to read from the bus. The 16 bit MAR is directly connected to the memory, The 16th bit decides whether the memory accessed is RAM or ROM. The RAM and ROM are each 32kb. When the 16th-bit is a 1, it selects the RAM. On power-up, the SDP-1C starts executing from address 0x0000. Inside the ROM is stored the BIOS of the processor, as well as the subroutines for communicating with the SDP-1G.

***3.1.4 The IDR***

The IDR of the SDP-1C consists of 1 byte, and is only allowed to read from the bus and store the value into the RAM.

***3.1.5 The ODR***

The ODR consists of 1 byte, and is only allowed to output onto the bus. The inputs of the ODR are either connected to the RAM or ROM depending on the value of the MAR. The ODR can either output onto the lower half of the bus or upper half.

***3.1.6 The PC***

The PC of the SDP-1C has a width of 2 bytes, and increments depending on a control line. The PC outputs its entire contents onto the 16-bit bus, and reads the entire 16-bit bus at a time.

***3.1.7 The SK***

The SK is separate from the RAM, and stores 16-bit values, and can output onto the bus, or read from the bus.

***3.1.8 The SP***

The SP can be incremented or decremented, and has an output of 8 bits which is directly connected to the input of the SK

***3.1.9 Th*e *IR***

The IR only takes an input from the bus, and the value of the IR is directly connected to the control logic.

***3.1.10 The Control Logic***

The control logic takes the value of the IR as an input, as well as the flags, and decides the signals to toggle based on the current instruction and flags.

***3.1.11 The IO Ports***

The SDP-1C has 2 input ports, and 2 output ports. The input ports are 8 bits each, and the output ports are also 8-bits each.

***The ISA of the SDP-1C***

| ***Instruction*** | ***Opcode*** | ***Operands*** | ***Flags set*** |  |
| --- | --- | --- | --- | --- |
| ldi | 0xc3 | Register, Value | No | 1 |
| adi | 0x89 | Register, Value | Yes | 2 |
| sbi | 0x8a | Register, Value | Yes | 3 |
| sub | 0x70 | Register, Register | Yes | 4 |
| add | 0x71 | Register, Register | Yes | 5 |
| xor | 0x88 | Register, Register | Yes | 6 |
| nand | 0x85 | Register, Register | Yes | 7 |
| jmp | 0x01 | Address | No | 8 |
| jc | 0x02 | Address | No | 9 |
| jnc | 0x03 | Address | No | 10 |
| jz | 0x04 | Address | No | 11 |
| jnz | 0x05 | Address | No | 12 |
| jm | 0x06 | Address | No | 13 |
| jp | 0x07 | Address | No | 14 |
| call | 0x50 | Address | No | 15 |
| rts | 0x51 | - | No | 16 |
| phi | 0x40 | Value | No | 17 |
| push | 0x41 | Register | No | 18 |
| pull | 0x42 | Register | No | 19 |
| move | 0x20 | Register, Register | No | 20 |
| load | 0x10 | Address, Register | No | 21 |
| store | 0x11 | Register, Address | No | 22 |
| dcr | 0xa0 | Register | Yes | 23 |
| inc | 0xa1 | Register | Yes | 24 |
| hlt | 0xff | - | No | 25 |
| nop | 0xfe | - | No | 26 |
| ror | 0xe0 | Register | Yes | 27 |
| rol | 0xe1 | Register | Yes | 28 |
| rri | 0xe2 | Value, Register | No | 29 |
| rli | 0xe3 | Value, Register | No | 30 |
| sfr | 0xe4 | Register | Yes | 31 |
| sfl | 0xe5 | Register | Yes | 32 |
| sli | 0xe6 | Value, Register | No | 33 |
| sri | 0xe7 | Value, Register | No | 34 |
| in | 0xb0 | Port-no, Register | No | 35 |
| out | 0xb1 | Register, Port-no | No | 36 |
| xri | 0xd0 | Value, Register | Yes | 37 |
| nandi | 0xd1 | Value, Register | Yes | 38 |
| sgr | 0xb8 | - | No | 39 |

**LDI**

The LDI instruction stands for Load Immediate which takes a value, and loads it into the corresponding register. For example, ldi 0x0f, A would load 0x0f into the A register, when assembled, it would look something like this - c3 0f 00.

**ADI And SBI**

The ADI and SBI stand for Add Immediate and Subtract Immediate. They Take a value, and add that with the selected register. For example, adi 0xea F would store the 0xea in the B register, and the F register into the A register, add the A and B registers, then store the result back into the F Register, and same thing for the SBI, but subtract instead of add.

**ADD And SUB**

The ADD and SUB instructions add or subtract the contents of 2 registers. For example,

add G D would add the contents of the G and D registers, and store the result in the G register. With the same applying the SUB, except it is subtraction instead of addition.

**XOR And NAND**

The XOR and NAND instructions XOR or NAND 2 registers. For example, xor G H would xor the values of the G and H registers, and store the result back in the G register. The same applies with NAND, except the result is the NANDed value of the 2 registers.

**Jumps**

The JMP instruction is an unconditional jump. It takes a 16-bit address to jump to as the operand. The rest of the jump instructions also take a 16-bit address to jump, and are described in Table-3

**CALL and RTS**

CALL is used to jump to a subroutine, and RTS is used to return from a subroutine. The CALL instruction takes the address of the subroutine to jump to as a parameter, and pushes the current address onto the stack, and jumps to the subroutine. The RTS instruction pulls the address of the main program from the stack into the program counter, and jumps to that address.

**PHI, PUSH And PULL**

The PHI instruction is used to push an immediate value onto the stack. The PUSH instruction is used to push the contents of a register onto the stack. PULL is used to pull the top of the stack into a register

***Table-3:***

| ***Jump Instruction*** | ***Description*** |
| --- | --- |
| jc | Jumps to the address if the carry flag is set |
| jnc | Jumps to the address is the carry flag is not set |
| jz | Jumps to the address if the zero flag is set |
| jnz | Jumps to the address if the zero flag is not set |
| jm | Jumps to the address if the sign flag is set |
| jp | Jumps to the address if the sign flag is not set |

**MOVE**

The MOVE instruction is used to move the contents of one register into another register. It takes 2 registers as operands.

**LOAD And STORE**

The LOAD instruction is used to load a value from memory into a register. It takes a memory address, and a register as operands. The STORE instruction stores the contents of a register into a memory location, and takes the location to store into and the register as operands.

**DCR And INC**

DCR is used to decrement the value of a register, and stores the result back into the register. It takes the register to decrement as an operand. The INC instruction is the same thing, but increments the register instead.

**HLT And NOP**

The HLT instruction is used to halt the computer and stop the clock, preserving its current state until power-off. NOP stands for No Operation. During the execution of a NOP, all register states are preserved, except the PC, MAR, IR, and ODR

**ROR And ROL**

ROR and ROL stand for Rotate Right and Rotate Left. These instructions take a register as a

parameter, and rotate it right or left once. The result of the rotation is stored back in the register that was rotated.

**RRI And RLI**

RRI and RLI stand for Rotate Right Immediate and Rotate Left Immediate. They take a value and register as operands, and rotate the value left or right once before storing the result into the specified register.

**SFR And SFL**

SFR and SFL stand for Shift Right, and Shift Left. They take a register as an operand, and shift the contents of the register left or right once, and then store the result back into the register.

**SLI And SRI**

SLI and SRI stand for Shift Left Immediate, and Shift Right Immediate. They take a value and register as operands, and shift the value left or right once, and then store the result into the specified register.

**IN And OUT**

The IN instruction takes an input port number and a register as operands. It takes the word at the specified input port number, and writes it into the specified register. The OUT instruction takes a register, and output port number as operands. It then writes the value of the specified register into the specified output port number. The OUT instruction also sets an acknowledge line high when the new data is in the port until the next clock cycle, then brings it low.

**XRI And NANDI**

XRI stands for XOR immediate, and takes a value, and a register as operands. It then XORs the value with the specified register, and then stores the result into the specified register. NANDI stands for NAND Immediate, and takes a value, and a register as operands. It then NANDs the value with the specified register, and stores the result into the specified register

**SGR**

SGR stands for Set GPU Render. It is a signal being read from the SDP-1G. When this instruction is executed, it will set the SGR signal high for one clock cycle, and then pull it low again.

***BLOCK DIAGRAM OF THE SDP-1C ARCHITECTURE:***



***3.2 SDP-1D***

The SDP-1D is a 4x4 LED matrix that is controlled by the SDP-1G. The cathode of all the LEDs are connected to each other, and the anode of each LED is connected to the SDP-1G. As there are 4x4 LEDs, the SDP-1G will need to have 16 output pins in order to interface with the SDP-1D.

***3.2.1 Visual Representation Of The SDP-1D From Top View:***

******

***3.2.3 Wiring***

The 16 output pins of the SDP-1G are mapped to the LED anodes. Pin 1 of the SDP-1G’s output is connected to the anode of LED 0 in the SDP-1D, and pin 16 of the SDP-1G’s output is connected to the anode of LED 15 in the SDP-1D.

***3.3 SDP-1G***

The SDP-1G interfaces with the output ports of the SDP-1C and the SGR signal, and outputs values for the SDP-1D to interpret. Port number 2 of the SDP-1C must be set to 0xff or 0xfe for the SDP-1G to read from port number 1 into its memory (VRAM).

***3.3.1 VRAM***

The SDP-1G has 2 bytes of VRAM. The VRAM is represented with two 8-bit registers. The inputs of the registers are connected to port number 1 of the SDP-1C. When port number 2 of the SDP-1C is set to 0xff, the SDP-1G will read the value of port number 1 into the 1st register as soon as the acknowledge line goes high. When port number 2 is set to 0xfe, the SDP-1G will read the value of port number 1 into the 2nd register when the acknowledge line goes high.

***3.3.2 The GL***

GL stands for GPU Latch. The GL is the last stage, and the outputs of the GL are what is connected to the anodes of the LEDs in the SDP-1D. The inputs of the GL are connected to the outputs of the VRAM. When the SGR signal is toggled high, the GL latches its inputs to the output.

***3.3.3 Visual Representation of the connections between the SDP-1x Modules***

******