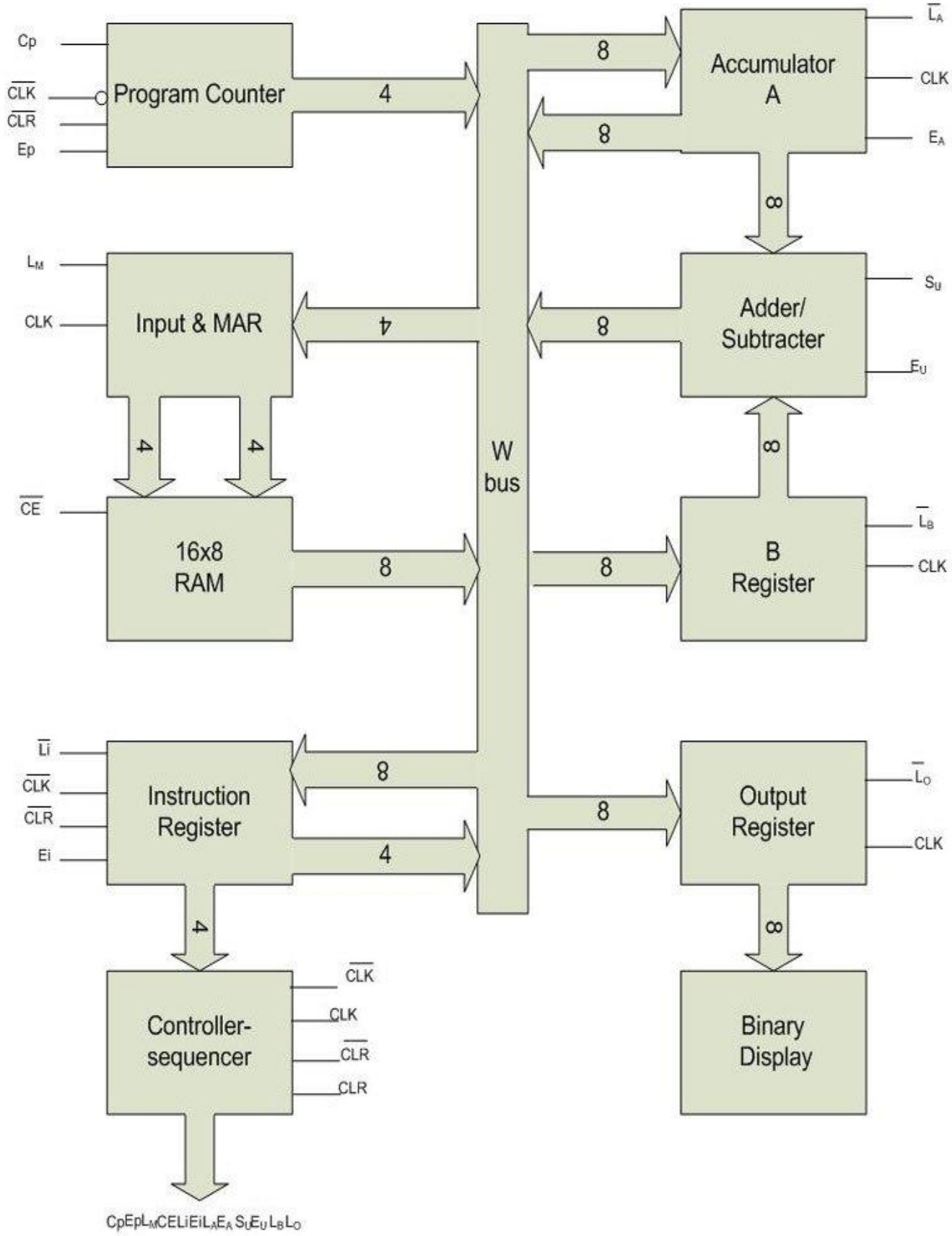
SAP-1 Architecture

The Simple-As-Possible (SAP)-1 computer is a very basic model of a microprocessor explained by Albert Paul Malvino. The SAP-1 design contains the basic necessities for a functional Microprocessor.

Its primary purpose is to develop a basic understanding of how a microprocessor works, interacts with memory and other parts of the system like input and output. The instruction set is very limited and is simple SAP (Simple-As-Possible)-1 is the first stage in the evolution toward modern computers.

Block Diagram of SAP-1



SAP is Simple-As-Possible Computer. The type of computer is specially designed for the academic purpose and nothing has to do with the commercial use. The architecture is 8 bits and comprises of 16 X 8 memory i.e. 16 memory location with 8 bits in each location, therefore, need 4 address lines which either comes from the PC (Program Counter which may be called instruction pointer) during computer run phase or may come from the 4 address switches during the program phase. All instructions (5 only) get stored in this memory. It means SAP cannot store program having more than 16 instructions.

AP can only perform addition and subtraction and no logical operation. These arithmetic operations are performed by an adder/subtractor unit.

There is one general purpose register (B register) used to hold one operand of the arithmetic operation while another is kept by the accumulator register of the SAP-1.

In addition, there are 8 LEDs which work as output unit and connected with the 8 bit output register.

All timely moment of data or activities are performed by the controller/sequencer part of the SAP-1.

Program Counter

- It counts from 0000 to 1111.
- It signals the memory address of next instruction to be fetched and executed.

Inputs and MAR (Memory Address Register)

- During a computer run, the address in PC is latched into Memory Address Register (MAR).

The RAM

- The program code to be executed and data for SAP-1 computer is stored here.
- During a computer run, the RAM receives 4-bit addresses from MAR and a read operation is performed. Hence, the instruction or data word stored in RAM is placed on the W bus for use by some other part of the computer.
- It is asynchronous RAM, which means that the output data is available as soon as valid address and control signal are applied.

Instruction Register

- IR contains the instruction (composed of OPCODE+ADDRESS) to be executed by SAP1 computer.

Controller-Sequencer

- It generates the control signals for each block so that actions occur in desired sequence. CLK signal is used to synchronize the overall operation of the SAP1 computer.
- A 12-bit word comes out of the Controller-Sequencer block. This control word determines how the registers will react to the next positive CLK edge.

Accumulator

- It is a 8-bit buffer register that stores intermediate results during a computer run.
- It is always one of the operands of ADD, SUB and OUT instructions.

Adder/Subtractor

- It is a 2's complement adder-subtractor.
- This module is asynchronous (unclocked), which means that its contents can change as soon as the input words change.

B-register

- It is 8-bit buffer register which is primarily used to hold the other operand (one operand is always accumulator) of mathematical operations.

Output Register

- This registers hold the output of OUT instruction.

Binary Display

- It is a row of eight LEDs to show the contents of output register.
- Binary display unit is the output device for the SAP-1 microprocessor.

SAP-1 Instructions and Instruction Cycle

SAP-1 instruction set consists of following instructions

Mnemonic	Operation	OPCODE
LDA	Load addressed memory contents into accumulator	0000
ADD	Add addressed memory contents to accumulator	0001
SUB	Subtract addressed memory contents from accumulator	0010
OUT	Load accumulator data into output register	1110
HLT	Stop processing	1111

The instruction format of SAP-1 Computer is

(XXXX) (XXXX)

The first four bits make the op-code while the last four bits make the operand (address).

Machine cycle and Instruction cycle

SAP1 has six T-states (three fetch and three execute cycles) reserved for each instruction. Not all instructions require all the six T-states for execution. The unused T- state is marked as No Operation (NOP) cycle. Each T-state is called a machine cycle for SAP1. A ring counter is used to generate a T-state at every falling edge of clock pulse. The ring counter output is reset after the 6th T-state.

FETCH CYCLE – T1, T2, T3 machine cycle

EXECUTE CYCLE - T4, T5, T6 machine cycle

- Complete code includes opcode and operand
- One instruction is executed in one instruction cycle
- Instruction cycle may consist of many machine cycles
- For SAP-1, Instruction cycle = Machine cycle
- Instruction cycle = Fetch cycle + Execution cycle
- Fetch cycle is generally same for all instructions
- Complete code includes opcode and operand
- Like LDA 04H 0000 0100
- One instruction is executed in one instruction cycle

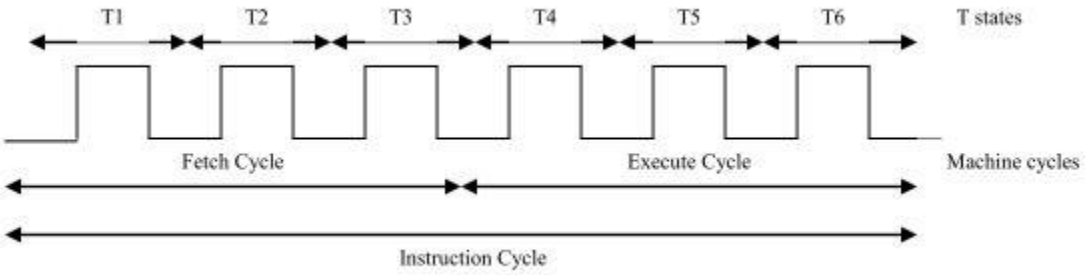


Fig: Instruction Cycle

Fetch and Execution Cycle of SAP-1 instructions

SAP-1 instruction cycle: 3 clock cycles to fetch and decode phase, 3 clock cycles to execute

The first three states are:

1. address
2. increment
3. memory

Controller has a 6-bit ring counter which continuously cycles from 000001 up to 100000 then resets (must be set to 000001 when we initialize the computer)

Ring counter is clocked on clock high-to-low transition, most of the other circuits in the computer on clock low-to-high transition

Fetch Cycle

Address state: enable PC to bus three-state output, MAR load line

Increment state: enable PC increment (and perhaps wait for memory access time)

Memory state: enable memory CE, IR load line

IR is loaded on the low-to-high clock transition, so stabilizes before state 4 is entered

t1: MAR ← PC

t2: PC ← PC +1

t3: IR ← RAM

Execution Cycle -- LDA

state 4: enable IR to bus three-state output, MAR load line

state 5: enable memory CE, accumulator load line

state 6: enable nothing

t4: MAR ← (IR (Address of operand))

t5: Accumulator ← RAM

t6: nothing

Execution Cycle -- ADD

state 4: enable IR to bus three-state output, MAR load line

state 5: enable memory CE, register B load line

state 6: enable add, ALU to bus three-state output, accumulator load line

t4: MAR ← (IR (Address of B))

t5: B ← RAM

t6: Accumulator ← Accumulator + B

Micro program

each SAP-1 block has some control lines:

- o each three-state driver has an enable line which connects the driver to the bus
- o the program counter also has a count line which, when high, increments the contents on the next low-to-high clock transition
- o all registers have a load line (active low)
- o the ALU has a subtract line which is high for subtraction and low for addition

implementing the SAP-1 instructions means raising and lowering these control lines at the appropriate times

these 12 control lines are the micro program word

controller must, on each clock cycle, produce 12 bits

some of these bits are on-off (e.g. three-state output lines) and have a "default off" state

some of these bits are A/B (e.g. the add-subtract line) and have a "default don't care"

some bits are active high, some are active low

Controller Implementation

How to generate micro words?

if a bit is only on during one cycle, connect it to the corresponding ring counter bit

if a bit is only on for an instruction, connect it to that instruction (as decoded by a 4-bit 1-of-16 decoder)

if a bit is only on for an instruction and a cycle, connect it to the AND of the ring counter bit and the decoder output

if a bit is on for multiple instruction and/or cycles, work out the truth table and use AND/OR or multiplexers to implement it

alternative: use a ROM

PART-II

Main features

Simple As Possible. One output device with 8 LEDs. 16 bytes of read only memory.
5 instructions
3 with 1 operand,
2 with implicit operands.
Accumulator Architecture
Accumulator, Out Register, **B Register**, Memory Address Register (**MAR**), Instruction Register (**IR**).

Architecture

8 bit "W" bus.
4-bit program counter, only counts up, it starts counting from 0 and counts up to 15.
4-bit Memory Address Register (MAR).
16 Byte Memory.
8-bit (1 Byte) Instruction Register (IR).
6-cycle controller with 12-bit microinstruction word.
8-bit Accumulator.
8-bit B Register.
8-bit adder/subtractor
8-bit Output Register.

Program counter

Instructions to be executed are placed at the starting addresses of memory, e.g. the first instruction of a program will be placed at binary address 0000 the second at address 0001.

Now to execute one instruction, first step is to generate the address at which this instruction is placed in memory.
So this address is generated by (4-bit) Program Counter, that counts from 0000 to 1111 (for total of 16 memory locations).
If the value of program counter is 0100, then the instruction at address at 4 will be executes next. program counter is like a pointer register; it points to the address of next instruction to be executed**.**

Input and Memory Address Register (MAR)

The MAR stores the (4-bit) address of data and instruction which are placed in memory.
When SAP-1 is Running Mode, the (4-bit) address is generated by the Program Counter which is then stored into the MAR through W bus.
A bit later, the MAR applies this 4-bit address to the RAM, where Data or instruction is read from RAM.
In Simulation we are using first 16 locations (0 to 15) of a 32x8 PROM.

The RAM

In initial design, the RAM is a 16 x 8 static TTL RAM. It means there are 16 memory locations (from 0 to 15) and each location contains an 8-bit of data/instruction.
You can program the RAM by means of the switches to be used for address and data. This allows you to store a program and data in the memory before a computer run
During a computer run , the RAM receives 4-bit addresses from the MAR and a read operation is performed,
in this way, the instruction or data stored in the RAM is placed on the W bus for use in some other part of the computer.

Instruction Register
When the instruction is placed at W-bus from memory, the Instruction Register stores this instruction on the next positive clock edge.
The contents of the instruction register are split into two nibbles.
The upper nibble is a two -state output that goes directly to the
block labeled "Controller-sequencer“
The lower nibble is a three-state output that is read onto the W-bus when needed.

Adder/subractor

SAP-1 uses a 2's complement adder-subtractor. When input Su is low (logic 0), the sum is: S = A + B
When S_u is high (logic 1), the sum is: S = A + B’ + 1
The Adder-subtractor is asynchronousand its contents change as soon as the input changes.
When E_U is high, these contents appear on the W bus.

Accumulator

To add/sub two 8-bit numbers A and B, the accumulator register stored the number A.
The Accumulator has two outputs.
One output goes to the adder/subtractor
The other goes to the W through tri-state buffers.

It also stores the (answer of two values) output of
adder/subtractor through w-bus, when LA is low.

It's value is appeared on w-bus when EAis high, which can then be read by output register.

B Register
To add/sub two 8-bit numbers A and B, the B register stored the number B.

It supplies the number to be added or subtracted from the contents of accumulator to the adder/subtractor.

When data is available at W-bus and L_b goes low, at the positive clock edge, B register loads that data.

Output Register
At the end of an arithmetic operation the accumulator
contains the word representing the answer,
Then answer stored in the accumulator register is then loaded into the output register through W-bus.
This is done in the next positive clock edge when E_A is high and L_O is low.
Now this value can be displayed to the outside world with the help of LEDs or 7 Segments.

Binary Display

The binary display is a row of eight light-emitting diodes (LEDs).
Because each LED connects to one flip-flop of the output port, the binary display shows us the contents of the output port.
Therefore, after we've transferred an answer from the accumulator to the output port, we can see the answer in binary form.
But we are using 7-segments in simulation.

Controller sequencer

The 12 bits coming out of the Controller Sequencer form a word that controls the rest of the computer. Before each operation a Clear (CLR) signal resets the computer.
The 12 wires carrying the control word are called the Control Bus. The control word has the format:

$$CON = \quad C_P \quad E_P \quad \overline{L_M} \quad \overline{CE} \qquad \overline{L_I} \quad \overline{E_I} \quad \overline{L_A} \quad E_A \qquad S_U \quad E_U \quad \overline{L_B} \quad \overline{L_O}$$

This word determines how the registers will react to the next positive clock (CLK) edge. For instance a high and a low means that the contents of Program Counter are latched into MAR on the next positive clocked age. As another example, a low and a low mean that the addressed RAM word will be transferred to the accumulator on the next positive clock edge**.**

Instruction set

- Computer is a useless hardware until it is programmed
- This means loading step-by-step instructions into the memory before the start of a *computer run*.
- Before you can program a computer, however, you must learn its *instruction set*, the basic operations it can perform. The SAP-1 instruction set follows.

SAP-1 INSTRUCTION SET		
Mnemonics	**Operation**	**Description**
LDA	ACC ← RAM[MAR]	Load RAM data into accumulator
ADD	ACC ← ACC + B	Add RAM data to accumulator
SUB	ACC ← ACC – B	Subtract RAM data from accumulator
OUT	OUT ← ACC	Load accumulator data into output register
HLT	CLK ← 0	Stop processing

- LDA stands for "load the accumulator," A complete LDA instruction includes the hexadecimal address of the data to be loaded.
- For example, LDA 8H means "load the accumulator with the contents of memory location 8H."
- Therefore, given RAM[8] = 1111 0000
- The execution of LDA 8H results in ACC = 1111 0000
- Similarly. LDA FH means "load the accumulator with the contents of memory location FH.

- ADD 9H means "add the data of memory location 9H with data of accumulator and save the result in accumulator.
- Suppose No. 2 is in the accumulator and No.3 is in memory location 9H. Then ACC =0000 0010, RAM[9] = 0000 0011
- During the execution of ADD 9H,
  - First data at RAM address 9 is loaded into the B register to get B = 0000 0011 and instantly the adder/subtracter forms the sum of A and B SUM = 0000 0101
  - Second, this sum is loaded into the accumulator to get ACC = 0000 0101
- Similarly, the execution of ADD FH adds data at RAM address 15 to the accumulator and save the answer back in accumulator overwriting the previous value.
- The negative numbers are stored in 2's complement form.

- SUB 9H means "subtract the data of memory location 9H from data of accumulator and save the result in accumulator.
- Suppose No. 3 is in the accumulator and No.2 is in memory location 9H. Then ACC =0000 0011, RAM[9] = 0000 0010
- During the execution of SUB 9H,
  - First data at RAM address 9 is loaded into the B register to get B = 0000 0010 and instantly the adder/subtracter forms the diff. of A and B Diff. = 0000 0001
  - Second, this diff. is loaded into the accumulator to get ACC = 0000 0001
- Similarly, the execution of SUB FH subtracts data at RAM address 15 from the accumulator and save the answer back in accumulator overwriting the previous value.

OUT Instruction

- The instruction OUT tells the SAP-1 computer to transfer the accumulator contents to the output port.
- After OUT has been executed, you can see the answer to the problem being solved on LEDs display.
- OUT is complete by itself; that is, you do not have to include an address when using OUT because the instruction does not involve data in the memory.

- HLT stands for halt. This instruction tells the computer to stop processing data so it stops the clock.
- HLT marks the end of a program, similar to the way a period marks the end of a sentence.
- You must use a HLT instruction at the end of every SAP-1 program; otherwise, you get computer trash (meaningless answers caused by runaway processing).
- HLT is complete by itself; you do not have to include a RAM word when using HLT because this instruction does not involve the memory.

Memory Reference Instructions

- LDA, ADD, and SUB are called *memory-reference instructions* because they use data stored in the memory.
- OUT and HLT, on the other hand, are not memory reference instructions because they do not involve the data stored in the memory.

Mnemonics

- LDA, ADD, SUB, OUT, and HLT are the instruction set for SAP-1. Abbreviated instructions like these are called *mnemonics* (memory aids). Mnemonics are popular in computer work because they remind you of the operation that will take place when the instruction is executed.

Op Codes of SAP-1

- To load instruction and data words into the SAP-1 memory , we have to use some kind of code that the computer can interpret.
- The number 0000 stands for LDA, 0001 for ADD, 0010 for SUB, 0000 for OUT, and 1111 for HLT.
- Because this code tells the computer which operation to perform, it is called an *operation code* (op code).
- *Assembly language* involves working with *mnemonics* when writing a program.
- *Machine language* involves working with strings of 0s and 1s.

TABLE 2, SAP-1 OP CODES

Mnemonics	Op Code
LDA	0000
ADD	0001
SUB	0010
OUT	1110
HLT	1111

Program in Assembly		Program in Machine Language		
Address	Contents	Address	Contents in Binary	Contents in Hexadecimal
0H	LDA 9H	0000	0000 1001	09H
1H	ADD AH	0001	0001 1010	1AH
2H	ADD CH	0010	0001 1100	1CH
3H	SUB BH	0011	0010 1011	2BH
4H	OUT	0100	1110 1111	EFH
5H	HLT	0101	1111 1111	FFH
6H	FFH	0110	1111 1111	FFH
7H	FFH	0111	1111 1111	FFH
8H	FFH	1000	1111 1111	FFH
9H	10H	1001	0001 0000	10H
AH	18H	1010	0001 1000	18H
BH	14H	1011	0001 0100	14H
CH	20H	1100	0010 0000	20H
DH	FFH	1101	1111 1111	FFH
EH	FFH	1110	1111 1111	FFH
FH	FFH	1111	1111 1111	FFH

The 8080 and 8085 Microprocessors

- The 8080 was the first widely used microprocessor.
- It has 72 instructions. The 8085 is an enhanced version of the 8080 with essentially the same instruction set (both are designed by Intel Corp.).
- The SAP-1 instructions are upward compatible with the 8080/8085 instruction set.
- In other words, the SAP-1 instructions LDA, ADD, SUB, OUT, and HLT are 8080/8085 instructions.
- Learning SAP instructions is getting you ready for the 8080 and 8085, two widely used microprocessors.