THE SDP-3 -- A COMPUTER FOR USE ON BOARD SMALL SCIENTIFIC SPACECRAFT

By: Mr. R. A. Cliff, Flight Data Systems Branch,
Spacecraft Technology Division,
NASA/Goddard Space Flight Center

1964

Abstract

The SDP-3 is a general-purpose stored-program computer which has been designed for use as the core of the data system of a small scientific spacecraft. It is a serial, two's complement, 16-bit-word machine. It has one level of indirect addressing and one hardware index register. To facilitate time-sharing operation there is a 16-level priority interrupt system and a real time clock. The 4K memory is divided into 16 pages of 256 words each. There is a user and a monitor mode. Inter-page access is prohibited in user mode.

Introduction

During the past decade general-purpose stored-program computers have become an integral part of most forms of scientific and technological endeavor. The omnipresence of computers results from their flexibility; the same machine may be used for many different applications by employing different programs. When a data handling or control system is needed it is usually cheaper and faster to use a mass-produced computer with a special program than to build a special-purpose system from scratch.

One of the most severe constraints on a small scientific spacecraft is overall weight. This occurs because launch vehicle costs increase with increasing payload. Unfortunately, the power system components, particularly batteries and solar panels are heavy. Therefore the weight constraint leads to a power constraint. As a result, the power available to the telemetry system is limited. This, in turn, limits the transmitted data rate no matter how efficient a code is used.

It is important to obtain as much data as possible from spacecraft because of their high cost. The data rate could be increased by using high gain antennas on the spacecraft, but this is difficult because of weight restrictions, aiming problems, etc. Another approach is to increase the worth of each transmitted data bit, rather than increasing the actual data bit rate. In other words, we shall achieve a worthwhile purpose if we can build some sort of processor which removes redundancy from the data, and which in so doing uses less power than would have been required to transmit the redundant data.

For a number of years now it has been customary to use special-purpose hard-wired data processors for the purpose of redundancy removal on board small scientific spacecraft. The problem is that the development of these special-purpose processors is expensive and time-consuming. If a stored-program computer were made an integral part of the spacecraft data system, then most special-purpose processors could be replaced by programs in the computer. Then, at the expense of some standardization, the difference between the data systems on many different spacecraft becomes chiefly a difference in computer programs.

There are a number of advantages to using a computer in a spacecraft data system. It becomes feasible, for instance, to have checked-out spacecraft on the shelf -- ready to have scientific experiments and programs inserted. This can significantly decrease the lead time from experiment concept to launch. If an experiment fails before launch, it can be replaced with a backup experiment. If the computer is also used to control the operation of the experiments, then in many cases it can prevent the generation of redundant data. This is more efficient than having to remove the redundant data from the data stream after it is generated. In addition, the memory of the computer can be used for buffering data before transmission. The format of the transmitted data can even be varied to suit conditions -- either by inserting new programs or by using a self-adaptive program.

This paper describes a new computer, the SDP-3, which has been designed for use as the core of the on-board data system of the proposed Advanced IMP spacecraft. In this application all data are collected and transmitted under computer control. First, however, we will fly the computer off-line on the IMP-I (eye) spacecraft (probably in MID 1970) as an engineering experiment. In this way we will verify the reliability of the system and gain experience with the operational aspects of using an on-board computer without compromising the success of a mission.

Design Ground Rules

The SDP-3 computer was designed to perform a number of functions in the Advanced IMP spacecraft. The first of these is programming of the scientific experiments. This includes sequencing operations in the experiment, controlling experiment modes, selecting sensor ranges, and calibration of the experiments.

The second function to be performed is data acquisition. In some cases, for instance randomly occuring cosmic ray events, this will be upon experiment request. Other measurements, such as magnetic field amplitude, are usually taken at uniformly spaced time intervals. IMP spacecraft are spin-stabilized; therefore data from directional sensors, for example plasma detectors, is most meaningful when sampled at uniform increments of spacecraft rotation.

Data compression includes -- among other techniques -- statistical analyses, interpolation, prediction, spectral analyses, etc. An adaptive format will be used to permit transmission of the most important data out of the total supply of data available. In conjunction with the adaptive format, a buffering function allows less important data to be held until there is time to transmit it during periods of low activity on the high-priority channels.

The computer must operate autonomously without intervention. It must, however, be possible to load portions of programs or entire programs from the ground in case the telemetered data shows that a different type of experiment programming or data processing would be advantageous.

Naturally, the computer must use as little hardware and as little power as possible. It must also be rugged and reliable.

General Description of the SDP-3

The organization of the SDP-3 computer reflects the above mentioned requirements. It is a 16-bit, serial, single word instruction, single address machine. It has 16 levels of priority interrupt and two modes -- user and monitor -- in order to effectively handle time-shared processing. The memory consists of up to 256 pages of 256 words each. A 16-page memory (4K word words) will be flown on IMP-I. Interpage access is prohibited in the user mode.

The SDP-3 features one level of indirect addressing and one hardware index register. Arithmetic is done in two's-complement fixed-point format. There is a shift-left-and-normalize instruction, but other floating-point operations and multiplication and division must be provided by software.

There is a continuously-running real-time clock which may be armed under program control to cause an interrupt at some selected time in the future. Also, the current value of the clock may be stored in the memory to mark the time of occurrence of some particular event. If there is a power outage, the computer marks its place and automatically restarts when power is restored. There are 32 binary control outputs which may be used to control experiments, the telemetry system, and the multiplexor which commutates the data from the experiments into the computer.

The SDP-3 (excluding the memory electronics) will consist of about 500 Fairchild LPDTuL circuits with an operating power dissipation of under two watts. Average dissipation will be even less because whenever the computer is idle, it enters a dormant low-power mode in which only the priority interrupt circuitry is energized. The volume is estimated to be of the order of 200 cubic inches and the weight about 4 pounds.

Figure 1 shows how the computer is divided into three functional units with serial data transfer between them. Each of these units will be a separate package in the spacecraft. The serial configuration reduces the number of potentially unreliable interconnections between the units. The internal organization of each unit is also serial whenever possible. This increases the reliability of the computer by drastically reducing the number of components required relative to a parallel machine. Concurrently one obtains important reductions in power drain, weight, and volume -- all at the expense of execution speed of course. The serial organization also makes it easier to incorporate modular redundancy at a register level.

Each SDP-3 instruction takes 64 bit times; this simplifies the control hardware. At the end of an arithmetic operation the result may either be put in the AC (in which case the second operand is restored in the memory) or the result may be stored in the memory location formerly occupied by the second operand (in which case the first operand remains unaltered in the AC).

The format of an instruction word is:

CCCCCC F T LLLLLLLL.

The 6-bit operation code CCCCCC allows 64 instructions of which 54 are being used. the flag bit, F, specifies indirect addressing if it is on and the tag bit, T, specifies indexing if it is on. Addresses within a page are specified by the 8-bit line number LLLLLLLL.

Words to be inserted into the memory under ground control are sent to the spacecraft via a PCM Command system. The SDP-3 has an input channel which contains two registers which hold respectively the transmitted word and the address at which to store it. When these registers have been filled from the ground (this takes about half a second) instruction execution is suspended for one instruction cycle while the new word is stored in the memory.

There is also an output channel which can continuously read out page 15 of the memory, one bit at a time, into the telemetry system. Whenever this channel needs another word from the memory, instruction execution is again suspended for one instruction cycle, just as it was with the input channel.

Addressing Scheme

Execution time in a serial computer increases with increasing word length. Power drain also increases as the registers get longer. Therefore a relatively short word length should be chosen. A spacecraft computer with a word length as short as 12 bits has been designed[1], but for the Advanced IMP more flexibility than this computer provides is required. Sixteen bits appears to be a reasonable compromise for the word length.

With a 16-bit word, and single word instructions, it is difficult to use many more than 8 bits for the address. Therefore the address field was chosen to be exactly 8 bits which is convenient because it is half of the total word length. Thus an instruction can reference one of only 256 memory locations. These are chosen to be 256 contiguous locations and constitute a page of memory. The first page starts at location 0, the second page at location 256, etc. An 8-bit page register holds the number of the page which is currently being used. There can be up to 256 pages of 256 words each for a total of 65,536 memory locations.

There is one 8-bit hardware index register. When indexing is specified, the contents of the index register are added (modulo 256) to the address field of the instruction. Thus even by indexing there is no way of referencing a location outside the page in which an instruction is located.

Indirect addressing, on the contrary, permits access to any location in the memory. When indirect addressing is specified, the first 8 bits of the referenced location are used to select the page and the second 8 bits are used to select the line within that page which is to be accessed. If indexing is also specified, the contents of the index register are then added (modulo 256) to the 8 bits which specify the line number.

A secondary advantage may be gained from the paged memory organization. In the spacecraft the computer must be time-shared between several programs. In order to keep these programs from being able to damage each other, one could put them in separate pages. Except for the indirect addressing feature they would then be unable to access one another. Accordingly two modes of operation are defined in the SDP-3. In the privileged or "monitor" mode indirect addressing functions exactly as described above. In the "user" mode, however, only the second 8 bits of the word accessed for an indirect address are used. These bits specify the line within the program's own page which is to be referenced. If a user program needs more than one page of memory, appropriate linkage between pages must be supplied by the time-sharing monitor program.

Timing

Timing in the SDP-3 is based on the bit clock which is derived from a crystal-controlled oscillator. The frequency of this oscillator is chosen to be a power of 2 times the desired telemetry system transmitted bit rate. We do this because this bit rate is derived from the computer when the computer feeds the transmitter directly. The bit period is 610 ns. on IMP-I.

Table 1. shows the various functions that occur during an instruction cycle and the amount of time that each takes. The total cycle of the computer is broken down into two sub-cycles -- the I (instruction) phase and the E (execution) phase.

1. R. A. Cliff "The SDP-1 Stored Program Computer" IEEE Trans. AES. To be published.

TABLE 1.

SDP-3 Timing

Function	Bit Times Required	
Address Instruction	8	
Fetch and Restore Instruction	8	
Address Indr. Adr. Loc. and Save Op. Code	8	
Fetch and Restore Indirect Address Location	8	
Index Arithmetic	8	
I Phase Total	40	40
Fetch Operand	4	
Execute Operation	16	
Restore Operand/Save Result	4	
E Phase Total	24	24
Total Instruction Cycle		64

During the I Phase the instruction is fetched and decoded and the effective address of the operand is computed. First, 8 bit times are used to transmit the line number and page number of the next instruction to the memory address register. Each of these numbers is sent serially; the two numbers are sent simultaneously. The next 8 bit periods are used to fetch and restore the instruction word. Then, during the third interval, the operation code is moved to the decoder and the address part of the instruction is moved to the line register in case indirect addressing is to be done. The fourth set of 8 bit periods is used to fetch and restore the indirect address if this has been requested; if not the computer is idle for 8 bit periods. During the last 8 bit periods of the I phase the effective address of the operand is computed and sent to the memory. If indexing has been specified it is done here; if not, the time still required to transmit the address to the memory address register.

During the E phase the operand (if any) is fetched, the operation specified by the operation code is performed, and then either the operand is restored in memory or the result of the operation is stored in memory. The read and restore cycles of the memory have been separated and the operation execution placed between them because in certain programs the ability to operate directly on a memory location is advantageous. In the usual case, in which the operand is restored in memory, this splitting produces no disadvantages.

In order to adequately cope with time-sharing and experiment sequencing, a real time clock is included in the SDP-3. This clock is a 16-bit register which counts instruction cycles. Each time this register overflows (every 65,536 instructions or about every half a second) a priority interrupt request is generated. Software then keeps a count of the number of overflows. There is a 16-bit time register which can be set under program control. A priority interrupt is also generated whenever the clock compares equal to this register.

Instruction Set

The instruction set for the SDP-3 is a compromise which has been chosen to make the execution of the required functions relatively efficient while not requiring too much hardware.

There are two kinds of instructions. One kind can only be executed in monitor mode. In user mode they cause execution of an instruction cycle during which no operation is performed. These instructions are the ones which manipulate the page register, the input/output hardware, or could otherwise cause trouble if used incorrectly.

The other kind of instruction (the majority) may be used in either mode. Execution of any of the 10 instructions whose operation codes have not been assigned causes the computer to execute an instruction cycle during which no operation is performed. Table 2. gives the 54 assigned operations.

The monitor mode instructions will be discussed first. The first of these, HALT, is used by the monitor program when there are momentarily no more tasks to perform. The HALT instruction turns off all power except that to the input channel, the output channel, the clock and the priority interrupt circuitry. If either the input channel or the output channel requests a memory access, the computer is turned on for one instruction cycle and then turns off again. Power is restored and the computer resumes execution of instructions upon receipt of a priority interrupt request.

The next three instructions are used for communication in either direction with the scientific experiments. Each one exchanges the contents of the specified memory location with a 16-bit register in one of the experiments. The particular experiment is selected by a multiplexer which is controlled by Control Register A. The three instructions allow sending execute pulses (which load or unload the register in the experiment) either before the exchange, after the exchange, or not at all.

The EINT instruction enables the interrupt system. The interrupt system is disabled by the DINT instruction. Likewise

TABLE 2.

SDP-3 Instruction List

Mnemonic	Operation
Monitor Mode Only	
HALT	Stop and turn off power
XMIA	Exchange memory with input, execute pulses after
XMIB	Exchange memory with input, execute pulses before
XMIN	Exchange memory with input, no execute pulses
EINT	Enable Interrupts
DINT	Disable Interrupts
ENOC	Enable Output Channel
DSOC	Disable Output Channel
LDCA	Load Control Register A
LDCB	Load Control Register B
LDMR	Load Interrupt Mask Register
LDTR	Load Time Register
XMPL	Exchange Memory With Page and Line Registers
Either Mode	
LINK	Subroutine and Interrupt Service Routine Linkage
CMAC	Compare Memory and AC
TUNC	Transfer Unconditionally
TDXR	Transfer and Decrement Index Register
TACZ	Transfer if AC = 0
TACM	Transfer if AC< 0
TMQE	Transfer if MQ Even
TOVF	Transfer if Overflow On
SIBO	Shift and Invert Bit Order
RLAQ	Rotate AC and MQ Left
SLAC	Shift Left AC
SRAC	Shift Right AC
SRAQ	Logical Shift Right AC and MQ
ARAQ	Arithmetic Shift Right AC and MQ
RRAC	Rotate Right AC
RRMQ	Rotate Right MQ
NORM	Floating Point Normalize AC and MQ
STAC	Store AC
STMQ	Store MQ
STXR	Store XR
STCL	Store Clock
STPL	Store Page and Line (Mark Place)
STSI	Store Sense Inputs
STZE	Store Zero
LDAC	Load AC
LDMQ	Load MQ
LDXR	Load XR
XMXR	Exchange Memory with XR
XMAC	Exchange Memory with AC
ADDA	Add to AC
ADDM	Add to Memory
SUBA	Subtract from AC, Result in AC
SUBM	Subtract from AC, Result in Memory
IORA	Or (Inclusive) to AC
IORM	Or (Inclusive) to Memory
ANDA	And to AC
ANDM	And to Memory
EORA	Exclusive Or to AC
EORM	Exclusive Or to Memory
MPAA	Multiple Precision Add to AC
MPSA	Multiple Precision Subtract from AC

the output channel is enabled by the ENOC instruction. It continues to run until disabled by a DSOC instruction.

Control Register A is used to control the input multiplexor, the experiment power switches, and other miscellaneous functions. It is set by the LDCA instruction and holds data until the next time it is loaded. The Control Register B is similar except that it is reset at the end of the instruction that sets it. Consequently the output line from a stage of this register will carry a 4 bit-time-long pulse each time that stage is set to a 1.

LDMR loads a mask register which can selectively disable any combination of interrupt levels.

The Time Register which is compared to the real-time clock is set by the LDTR instruction.

XMPL is a mark place and branch instruction which is used for subroutine linkage within the monitor.

The first group of instructions which may be executed in either mode is made up of branching instructions. The first of these, LINK, exchanges the page register, line register, index register, carry flip-flop, and interrupt inhibit flip-flop with two words in memory. In monitor mode the address field of the link instruction is taken modulo 32, in user mode it is taken modulo 16. The instructions LINK 16 through LINK 31 are identical in execution to the servicing of interrupt requests on levels 0 through 15 respectively. This instruction is explained more fully in the section of this paper on the priority interrupt system.

CMAC skips the next sequential instruction if the referenced memory location and the AC are unequal. It is used for searching tables.

TDXR transfers and decrements the index register by one if the index register is non-zero. If the index register is zero, this instruction has no effect. It is used for controlling loops.

The other transfer instructions are self-explanatory.

The next set of instructions is used for shifting data in the AC and MQ. Quite a number of these are included to facilitate the various formatting functions the computer must perform. The SIBO instruction is of particular interest. It causes bits to shift right in the MQ, out of the right end of the MQ, into the right end of the AC and then left in the AC. One particular application of this instruction is inverting the order of the bits which have been generated by an analog-to-digital converter. When a successive approximation converter is used, the bits are generated most significant bit first. In order to do arithmetic on the data in the computer, it must first be turned around.

The NORM instruction shifts the AC and MQ left until the AC contains a normalized fraction. It simultaneously increments the referenced memory location by the number of shifts required to bring the AC into normalized form.

The next group of instructions is used for storing and loading the various registers. On the whole, they are self-explanatory. STSI stores the state of 16 input lines which are available for sensing external conditions.

The arithmetic instructions appear in the last group. They include addition, subtraction, and, inclusive or, and exclusive or. The multiple precision operations function identically to the other addition and subtraction operations except that the carry flip-flop is not initialized beforehand. Thus whatever carry was left over at the end of the last add or subtract is added to the result of the operation. It is expected that multiple precision arithmetic will normally be used in preference to floating-point arithmetic because of its speed. Then conversion to floating-point form will be done just prior to transmission of the data.

Priority Interrupt System

The SDP-3 has 16 priority interrupt lines. During the I phase of each instruction these lines set a shift register. Then the register is shifted until a one is detected at its output. If a one is detected, the number of shifts required to produce it is equal to the level of the highest priority active request. If no one is detected, the computer continues its operation undisturbed. If a one is detected, the computer finishes the instruction which it is executing. Then it automatically executes a 'LINK L + 16" instruction (where L is the interrupt level). The link instruction saves the carry, overflow, mode, and interrupt inhibit flip-flops and the index register in page 0, line L + 16. It simultaneously loads these registers from this location with the values that they last had during the level L interrupt service routine. The page number and line number at which to resume operation of the interrupted program are stored in page 0, line L + 48. Simultaneously the computer picks up the address of the first instruction of the interrupt service routine from this location.

An interrupt service routine to fetch one word of data from an experiment and store it in a buffer area within a user program is given below. Each interrupt request stores one word of data, until the buffer is full. Further requests are then ignored.

```
ISR4    LINK    4 + 16
        LDCA    ADR4
        XMIN I X BUFR4
        TDXR    ISR4

        LDMR    M4
        TUNC    ISR4
BUFR4   DAC     (Page and Line of buffer area)
 M4     DC      /0100
ADR4    DC      /4000
```

The first word of the program to be executed is the LDCA. This is caused by properly initializing page 0, line 52 (4 + 48) when the computer is loaded. The LDCA instruction loads a code into the Control Register A which selects the desired experiment. Page 0, line 20 (4 + 16) is initialized with the interrupt inhibit bit on, the monitor mode on, and the index register bits set to the number of words in the buffer area. Consequently the computer is in monitor mode, and the index register contains the number of words in the buffer when the XMIN is executed. This instruction stores the input data indirectly in the last word of the buffer area.

Then the TDXR instruction decrements the index register and transfers to the LINK instruction. The LINK instruction saves the monitor mode and interrupts disabled states of the interrupt service routine and also saves the new value of the index register. In addition it saves the address of the next sequential instruction -- the first instruction of the interrupt service routine. Simultaneously the LINK instruction restores the computer to the state it was in before it started the interrupt service routine. The next instruction to be executed is the one which would have been next in sequence in the original program if the interrupt service routine had not been initiated. A total of 5 instruction cycles of delay has occurred to this program.

As interrupt requests are received, the interrupt service routine works backward through the buffer area, one word at a time. When the buffer area is full, the TDXR instruction will mask out the level 4 interrupt. Then control is returned to the interrupted program through the LINK instruction. All further requests for service on level 4 will be ignored until the mask register is reset (after the buffer has been emptied).

Conclusion

The detailed design of this computer is now nearly complete. It has been simulated on our IBM 1800 computer. We also have an assembler running on the IBM 1800. We are now constructing a breadboard of the computer and working on the time-sharing monitor program which will be described in a future paper.

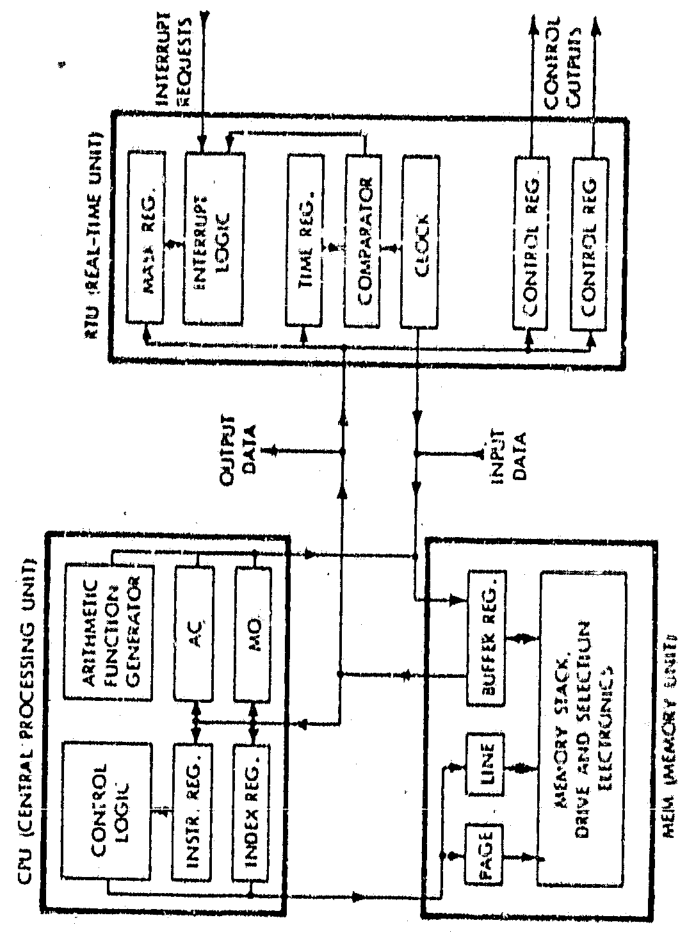

Figure 1 - Functional Hardware Grouping