Microcomputer design

6 — The Z80 microprocessor explained

by Phil Pittman, B.Sc. in association with NASCO Ltd

Having considered some of the generalities of microcomputer hardware and software in previous articles, and also some parts of a particular practical system, this six-part series concludes by looking more closely at the central processing unit — in this case the Z80 microprocessor. Although some of the following information has been given in previous articles (November and December 1977; January, February and August 1978), it is being repeated here for completeness and greater detail.

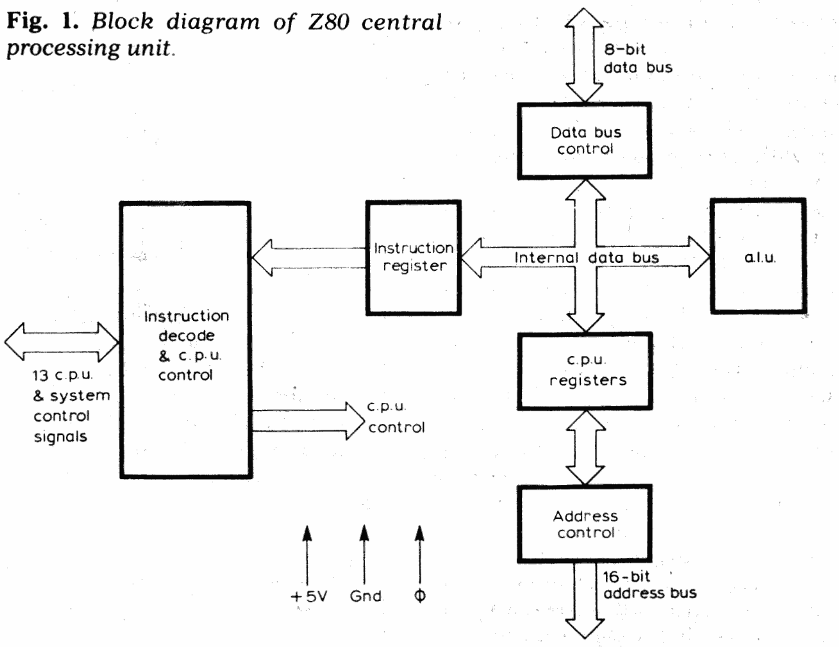
A BLOCK DIAGRAM of the internal architecture of the Z80 central processing unit is shown in Fig. 1. The diagram shows the major elements in the c.p.u. and it should be referred to throughout the following description. First let us look at the c.p.u. registers. The Z80 c.p.u. contains 208 bits of read/write memory that are accessible to the programmer. Fig. 2 illustrates how this memory is arranged into eighteen 8-bit registers and four 16-bit registers. The registers include two sets of six general purpose registers that may be used individually as 8-bit registers or in pairs as 16-bit registers.

Special purpose registers

1. Programme counter (p.c.). The programme counter holds the 16-bit address of the current instruction being fetched from memory. The p.c. is automatically incremented after its contents have been transferred to the address lines. When a programme jump occurs, the new value is automatically placed in the p.c., overriding the incrementer.

2. Stack pointer (s.p.). Any portion of external r.a.m. may be dedication as a stack area. This is used as a method of sequentially storing or retrieving data on a last-in first-out (l.i.f.o.) basis. The s.p. holds the 16-bit address of the current top of stack. Data can be "pushed" onto the stack, 16-bits at a time, from specific c.p.u. registers or "popped" off the stack into specific c.p.u. registers through the execution of PUSH and POP instructions. The data popped from the stack is always the last data which was pushed onto it. Any stack push or pop automatically modifies the s.p. in such a way that the s.p. always contains the address of the current top of stack. The stack is frequently used to save

Fig. 1. *Block diagram of Z80 central processing unit.*

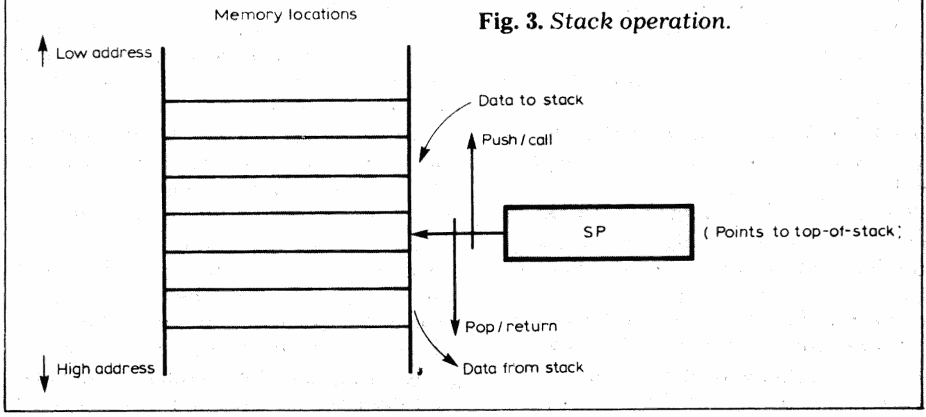


Main register set		Alternate register set		
Accumulator A	Flags F	Accumulator A'	Flags F'	
B	C	B'	C'	General purpose registers
D	E	D'	E'	
H	L	H'	L'	

interrupt vector I	Memory refresh R	Special purpose registers
Index register IX		
Index register IY		
Stack pointer SP		
Program counter PC		

Fig. 2. *Arrangement of registers in the Z80 c.p.u.*

Fig. 3. *Stack operation.*



programme counter contents before certain types of jumps (calls) so that the programme can later return to the same place again by popping the old value back to the p.c.

The stack allows simple implementation of multiple level interrupts, unlimited subroutine nesting and simplification of many types of data manipulation. Fig. 3 indicates the operation of the stack.

3. Two index registers (IX and IY): the two independent index addressing modes. An index register is used as a base to point to a region in memory in which data is to be stored or from which it is to be retrieved. An additional byte is included in indexed instructions to specify a displacement, either positive or negative, from this base. This mode of addressing greatly simplifies many types of programme, especially where tables of data are used.

4. Interrupts page address register (I). The Z80 c.p.u. can be operated in a mode where an indirect call (a special type of jump) to any memory location can be achieved in response to an interrupt. The I register is used for this purpose to store the high order 8-bits of the memory of the address. This feature allows the interrupt service programme to be located anywhere in memory with absolute minimal access time to the routine.

5. Memory refresh register (R). The Z80 c.p.u. contains a memory refresh counter to enable dynamic memories to be used with the same ease as static memories. While a discussion of dynamic r.a.m. is beyond the scope of this article, it is sufficient to say that dynamic r.a.m. stores its data as charges on capacitors. In order that this charge does not decay it is necessary to provide a partial address for the blocks of memory cells, plus certain clock pulses, within a specified minimum time. These are the functions provided by the Z80. The 7-bit refresh register is automatically incremented after each instruction fetch. The data in the refresh counter is sent out on the lower portion of the address bus along with a refresh control signal while the c.p.u. is decoding and executing the fetched instruction. This mode of refresh is totally transparent in that it does not slow down the c.p.u. operation. The programmer can load the register for testing purposes, but this register is not normally used by the programmer.

Accumulator and flag registers

The c.p.u. includes two independent 8-bit accumulators and associated 8-bit flag or status registers. The accumulator holds the results of 8-bit arithmetic or logical operations while the flag or status register indicates specific conditions for 8- or 16-bit operations, such as indicating whether or not the result of an operation is equal to zero. The programmer selects the accumulator and flag pair with which he wishes to work with a single exchange instruction so that he may easily work with either pair.

General purpose registers

There are two matched sets of general purpose registers, each set containing six 8-bit registers that may be used individually as 8-bit registers or as 16-bit register pairs by the programmer. One set is called BC, DE and HL while the complementary set is called BC′, DE′, and HL′. At any one time the programmer can select any one set to work with through a single exchange command for the entire set. In systems where a fast interrupt response is required, one set of general purpose registers and an accumulator/flag register may be preserved for handling this very fast routine. Only simple exchange instructions need be executed to go between the routines. This greatly reduces interrupt service time by eliminating the requirement for saving and retrieving register contents in the external stack during interrupt or subroutine processing. These general purpose registers are used for a wide range of applications by the programmer.

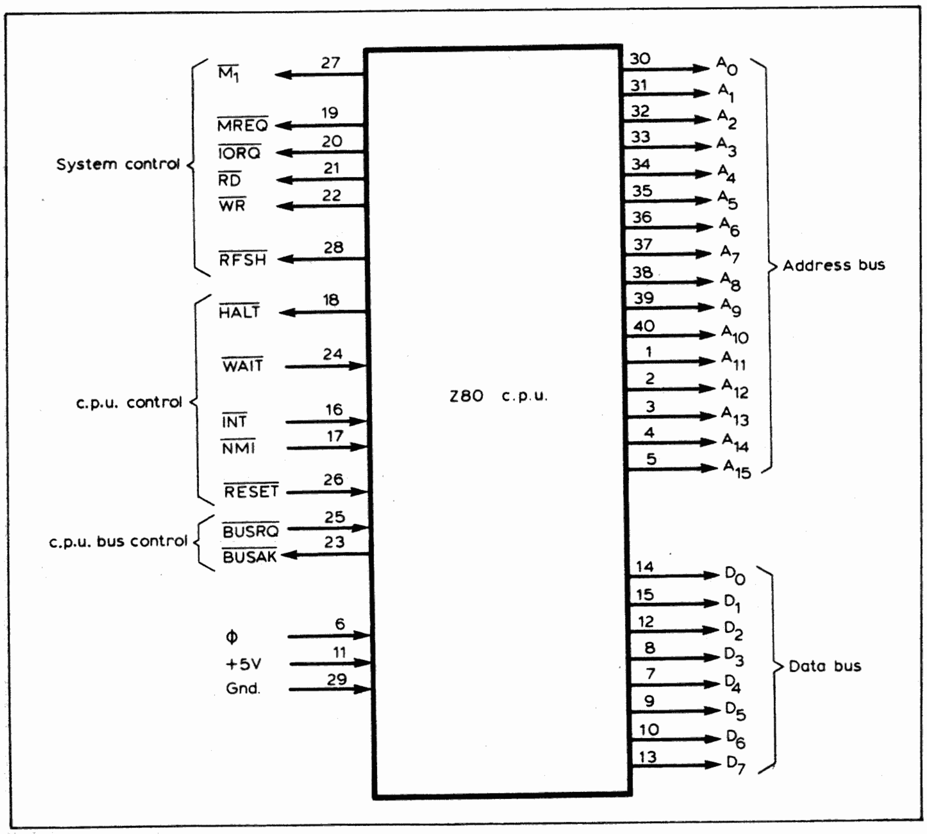


Fig. 4. *Pin numbers and functions in the Z80. The abbreviations are explained in the text.*

Arithmetic and logic unit (a.l.u.)

The 8-bit arithmetic and logical instructions of the c.p.u. are executed in the a.l.u. Internally the a.l.u. communicates with the registers and the external data bus or the internal data bus. The type of functions performed by the a.l.u. include: add, subtract, logical AND, logical OR, logical exclusive OR, compare, left or right shifts or rotates, increment, decrement, set bit, reset bit, and test bit.

Instruction register and c.p.u. control

As each instruction is fetched from memory, it is placed in the instruction register and decoded. The control section performs this function and then generates and supplies all of the control signals necessary to read or write data from or to the registers, control the a.l.u. and provide all required external control signals.

External signals

The Z80 is a single chip c.p.u. packaged in a standard 40-pin dual-in-line package. Fig. 4 shows the functions which are brought out to the external pins of the device while Fig. 5 shows how the device fits into the microcomputer circuit. All outputs from the c.p.u. with the exception of $\overline{\mathrm{M1}}$, $\overline{\mathrm{RFSH}}$, $\overline{\mathrm{HALT}}$ and $\overline{\mathrm{BUSAK}}$ have a three-state capability. With the exception of the data and address buses all signals have an active low state. The following paragraphs explain the various signals and connections shown as code names in Fig. 4 and Fig. 5.

Address bus (A_0-A_{15}). Pins A_0-A_{15} constitute a 16-bit address bus. The bus provides the address for memory (up to 64K bytes), data exchange and for i/o device data exchanges. I/o addressing uses the eight lower address bits to allow the user directly to select up to 256 input or 256 output ports. A_0 is the least significant address bit. During refresh time, the lower seven bits contain a valid refresh address.

Data bus (D_0-D_7). The 8-bit bidirectional data bus is used for data exchanges with memory and i/o devices.

Machine cycle one ($\overline{\text{M1}}$). This indicates that the current machine cycle is an instruction fetch cycle.

Memory request ($\overline{\text{MREQ}}$). The memory request signal indicates that the address bus holds a valid address for a memory read or memory write operation.

Input/output request ($\overline{\text{IORQ}}$). The input-output request signal indicates that the lower half of the address bus holds a valid i/o address for an i/o read or write operation. An $\overline{\text{IORQ}}$ signal is also generated with an $\overline{\text{M1}}$ signal when an interrupt is being acknowledged to indicate that an interrupt response vector (address) can be placed in the data bus by the interrupting peripheral. Interrupt acknowledge operations occur during M1 time while i/o operations never occur during M1 time.

Read ($\overline{\text{RD}}$). The "read" pulse indicates that the c.p.u. wants to read data from memory or an i/o device. The addressed i/o device or memory should use this signal to gate data onto the c.p.u. data bus.

Write ($\overline{\text{WR}}$). The "write" signal indicates that the c.p.u. data bus holds valid data to be stored in the addressed memory or i/o device.

Refresh ($\overline{\text{RFSH}}$). The "refresh" signal indicates that the lower seven bits of the address bus contain a refresh address for dynamic memory and the current $\overline{\text{MREQ}}$ signal should be used to do a refresh operation on all dynamic memories.

Halt state ($\overline{\text{HALT}}$). The $\overline{\text{HALT}}$ output from the c.p.u. indicates that a "halt" software instruction has been executed. The c.p.u. remains halted until reset or interrupted. During a halt, refresh activity is maintained.

Wait ($\overline{\text{WAIT}}$). The "wait" input may be used to indicate to the c.p.u. that the addressed memory or i/o devices are not ready for a data transfer. Additional one clock cycle timing states are generated for as long as the "wait" signal is active. This signal allows any speed of memory or i/o device to be synchronised to the c.p.u.

Interrupt request ($\overline{\text{INT}}$). The "interrupt request" signal is generated by i/o devices. A request will be honoured at the end of the current instruction if the internal software controlled interrupt enable flag is enabled. When the c.p.u. accepts the interrupt, an acknowledge signal ($\overline{\text{IORQ}}$) during M1 time) is sent out at the beginning of the next instruction cycle. The c.p.u. can respond to an interrupt in three different modes that are selected by software instructions.

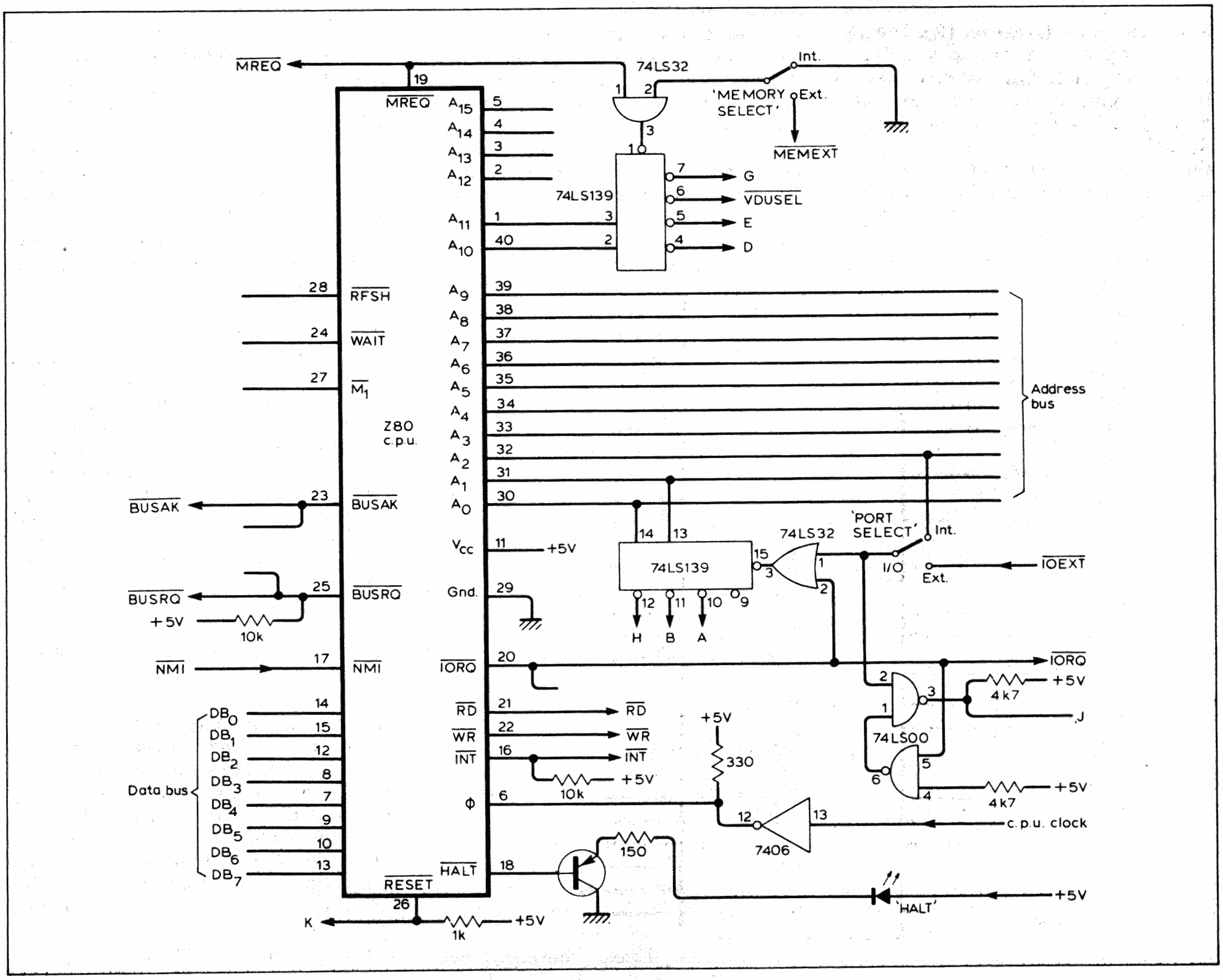
Non-maskable interrupt ($\overline{\text{NMI}}$). The non-maskable interrupt request line has a higher priority than $\overline{\text{INT}}$ and is always recognised at the end of the current instruction, independently of the status of the interrupt enable flag. $\overline{\text{NMI}}$ automatically forces the Z80 to restart at memory address 0066 hex. The programme counter is saved automatically in the external stack so that the user can later return to the programme that was interrupted.

Reset ($\overline{\text{RESET}}$). A reset forces the programme counter to zero and initialises the c.p.u.

Bus request ($\overline{\text{BUSRQ}}$). The "bus request" signal is used to request the c.p.u. address bus, data bus and three-state output control signals to go to a high impedance state so that other devices can control these buses. The request will be granted as soon as the current c.p.u. machine cycle is completed.

Bus acknowledge ($\overline{\text{BUSAK}}$). "Bus acknowledge" is used to indicate to the

Fig. 5. *How the Z80 m.p.u. is used in the microcomputer, showing address and data buses and other associated logic.*



requesting device that the c.p.u. address bus, data bus and control bus signals have been set to their high impedance state and the external device, e.g. the d.m.a. controller, can now control these buses.

Clock (I). The Z80 c.p.u. requires a single phase t.t.l. square wave clock for timing control. The frequency of this is 2.5 MHz for the standard Z80 or 4.0 MHz for the Z80A.

As shown in Fig. 5 the c.p.u. clock is driven from a conventional t.t.l. buffer with a 330Ω pull-up resistor, as required by the package. The input to the buffer can be selected from points on the video r.a.m. frequency divider chain (August issue, p.56), which is driven from a 16MHz crystal. A link has been provided to allow the clock frequency to be set to 1, 2 or 4MHz.

The logical design of the microcomputer has been arranged to exploit the non-maskable interrupt facility of the Z80, mentioned above, for a very special purpose – to provide a single step action for programme development work. By utilising external logic to interrupt the processor a fixed number of M1 cycles after a known command has been executed, the execution of the programme can be halted by causing the interrupt to occur during a particular instruction. The software arranges successive instructions in a programme to be interrupted, and immediately after the interruption all major registers are mapped into the video r.a.m. (August 1978 issue) and are consequently displayed on the tv set. The software will then wait for a specific keystroke to move the next instruction into the interrupting position. All this is necessary because instructions can be of differing lengths, and unless a huge search table is provided to establish the lengths of each of the 158 different Z80 instructions, the software has no other means of "knowing" which bytes are instructions, which are data and which are operands.

The particular method chosen for this system is to cause a non-maskable interrupt on the fourth M1 cycle after the low-to-high transition of bit 3 of port 0. The counting and blocking of the interrupt is performed by 74LS74 integrated circuits and associated gates. This system is also reset by the c.p.u. reset signal.

The instruction set

A previous article in this series has outlined the various groups of instructions constituting the Z80 instruction set. The following paragraphs provide more detail on the facilities offered by the instruction set, although it is not possible to give full descriptions in the space available.

Load and exchange. These are the main instructions used for transferring data around the system between registers and memory locations. Any 8-bit quantity may be freely moved around by utilising one of a variety of addressing modes with the basic "load" (LD) instruction. Register to register transfers are the simplest but in a register-to-memory or memory-to-register operation the memory address may be provided in one of a number of ways. The data may be part of an instruction, in which case it is fetched from the programme memory in the normal way. Alternatively, a 16-bit data memory address may be provided as part of the instruction. A common method of addressing data memory with the Z80 is to make use of the various 16-bit registers (IX, IY, BC, DE, HL) to contain data addresses.

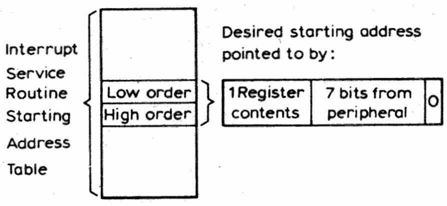


Fig. 6. *Interrupt response.*

Sixteen-bit quantities may also be transferred by the Z80 in a single instruction. For example 16-bit data to be placed in a 16-bit register may be included as part of the instruction, or the first address at which a 16-bit register pair of values is to be transferred between may be included in an instruction. Also any 16-bit register (excluding s.p.) may be pushed or popped on or off of the external stack. Exchange instructions allow the selection of either register bank or accumulator and also include various 16-bit register swaps.

Block transfer and search. An extremely powerful set of block transfer instructions exist in the Z80 for moving a block of data of any size from one memory area to another in a single instruction. All of these instructions operate with three registers: HL points to the source location, DE points to the destination location, BC is a byte counter. After the programmer has initialised these registers a single instruction can transfer a byte from the location pointed to by HL to the location pointed to by DE. These two points are then either incremented or decremented depending on the instruction being used, and the byte counter BC is decremented. The next byte is then transferred, and so on until BC = 0.

The block search instructions will search a given memory block for a specific data value with a single instruction. Again HL is used as a memory address pointer and BC is used as a byte counter. The accumulator is used to contain a copy of the value to be searched for. When the search instruction is executed it will sequence through the memory, updating HL and BC until it either finds a match with the accumlator contents or BC reaches zero.

Arithmetic and logical. The Z80 is capable of performing a wide range of 8-bit and 16-bit arithmetic and logical operations, as listed earlier. In all of these instructions except increment and decrement, the specified 8-bit operation is performed between the data in the accumulator and the specified source of data. This source may be any of the c.p.u.'s 8-bit registers, memory address by (HL), (IX + d) or (IY + d) or "immediate" data contained as part of the instruction. The result of the operation is placed in the accumulator, with the exception of the "compare" instruction which leaves the accumulator unaffected. All of these operations affect the flag register as a result of the specified operation.

The facilities of the flag register and instruction set allow arithmetic operations for multiprecision b.c.d. numbers, multiprecision signed or unsigned binary numbers, and multiprecision two's complement signed numbers.

A group of 16-bit arithmetic instructions allow various operations between the Z80's 16-bit register, frequently using HL as a 16-bit accumulator. These simplify address calculations or other 16-bit arithmetic operations.

Bit manipulation. The ability to set, reset or test individual bits in a register or memory location is needed in almost every programme. These bits may be flags in a general purpose software routine, indications of external control conditions or data packed into memory locations to make memory utilisation more efficient.

The Z80 has the ability to set, reset or test any bit in the accumulator, any general purpose register or any memory location with a single instruction.

Jump, call and return. A "jump" is a branch in a programme where the programme counter is loaded with the 16-bit value specified by one of a number of available addressing modes. The "jump" group has several different conditions that can be specified to be met before the jump will be made. If these conditions are not met, the programme merely continues with the next sequential instruction. The conditions are all dependent on the data in the flag register. Jump addresses may either be determined from information contained as part of the instruction or from certain of the c.p.u.'s 16-bit registers. The latter capability allows programme jumps to be a function of previous calculations.

A "call" is a special form of jump where the programme counter contents are pushed onto the stack (addresses by the stack pointer register) before the jump occurs. A "return" is the reverse of a "call", in that the value on top of the stack is popped directly into the p.c. to form a jump address. The "call" and "return" allow for easy handling of subroutines and interrupts.

Input/output. The transfer of data between the microcomputer and the peripheral devices is accomplished via the c.p.u. 8-bit registers with the aid of instructions from the i/o group. An eight-bit port address may be specified either as part of the instruction or as the contents of register C. Special block i/o instructions of the Z80 allow the transfer of complete blocks of data directly between an i/o port and memory with a single instruction similar to those for block memory moves.

Flags. Each of the two Z80 c.p.u. flag registers contains six bits of information which are set or reset by various c.p.u. operations. Four of these bits are testable; that is, they are used as conditions for jump, call or return instructions. For example, a jump may be desired only if a specific bit in the flag register is set. The four testable flag bits are:

1. *Carry flag (C).* This flag is the carry from the highest order bit of the accumulator. For example, the carry flag will be set during an add instruction where a carry from the highest bit of the accumulator is generated. This flag is also set if a borrow is generated during a subtract instruction. The shift and rotate instructions also affect this bit.

2. *Zero flag (Z).* This flag is set if the result of the operation loaded a zero into the accumulator. Otherwise it is reset.

3. *Sign flag (S).* This flag is intended to be used with signed numbers and is set if the result of the operation was negative. Since bit 7 represents the sign of the number (a negative number has a 1 in bit 7), this flag stores the state of bit 7 in the accumulator.

4. *Parity/overflow flag (P/V).* This dual purpose flag indicates the parity of the result in the accumulator when logical operations are performed, and it represents overflow when signed two's complement arithmetic operations are performed. The Z80 overflow flag indicates that the two's complement number in the accumulator is in error since it has exceeded the maximum possible (+127) or is less than the minimum possible (−128) number that can be represented in two's complement.

There are also two non-testable bits in the flag register. Both of these are used for b.c.d. arithmetic. The "half carry" (H) flag is the b.c.d. carry or borrow from the least significant four bits of the a.l.u. This is examined by the Z80's special "decimal adjust accumulator" instruction used when performing decimal arithmetic. The "subtract flag" (N) is also used by the decimal adjust instruction to indicate if the previous arithmetic instruction was an addition or subtraction.

The flag register can be accessed by the programmer and has the following format:

S	Z		H		P/V	N	C

Interrupt response. The purpose of an interrupt is to allow peripheral devices to suspend c.p.u. operation in an orderly manner and force the c.p.u. to start a peripheral service routine. Usually this routine is involved with the exchange of data or status and control information, between the c.p.u. and the peripheral. Once the service routine is completed, the c.p.u. returns to the operation from which it was interrupted.

The Z80 has two interrupt inputs, a software maskable interrupt and a non-maskable interrupt. The non-maskable interrupt (n.m.i.) cannot be disabled by the programmer and it will be accepted whenever requested by a peripheral device. This interrupt is generally reserved for very important functions that must be serviced whenever they occur, such as impending power failure. When the Z80 receives a non-maskable interrupt it performs an automatic subroutine call to a predetermined memory address (0066 hex).

The maskable interrupt (INT) can be selectively enabled and disabled by the programmer. This allows the programmer to disable the interrupts during periods where his programme has timing constraints that do not allow it to be interrupted. The Z80 can be programmed to respond to maskable interrupts in any one of three possible modes.

Since the Z80 was evolved from the 8080A microprocessor, i.e. the 8080A's instruction set and internal register organisation is a sub-set of the Z80's, one of the Z80's interrupt modes is identical to that of the 8080A. In this mode, when the c.p.u. acknowledges an interrupt, it expects some external hardware to supply an instruction to the data bus. The c.p.u. then executes this (usually a jump or call) rather than getting the next instruction from the programme memory. This means that an 8080A can easily be replaced by a Z80 in a system without necessarily modifying the interrupt system, especially as 8080A programmes are upward compatible, at the binary machine code level, with the larger Z80 instruction set.

For simple interrupt requirements the second mode of Z80 interrupt response is quite attractive. In this mode, whenever an interrupt is accepted the c.p.u. performs an automatic subroutine call to a predetermined address (0038 hex).

The third mode of Z80 interrupt response is the most powerful. In this mode the interrupting device is required to identify itself by supplying an 8-bit number (vector) to the c.p.u. when the interrupt is acknowledged. (Note that the Z80 activates both $\overline{\mathrm{M1}}$ and $\overline{\mathrm{IORQ}}$ simultaneously to signify an interrupt acknowledge cycle.)

With this mode the programmer maintains a table of 16-bit starting addresses – one for every interrupt service routine. The table may be located anywhere in memory. When an interrupt is accepted, a 16-bit pointer must be formed to obtain the desired interrupt service routine starting address from the table. The upper eight bits of this pointer are formed from the contents of the c.p.u.'s I register, which must have been previously set up by the programmer. The lower eight bits of the pointer are supplied by the interrupting device. Using the pointer to the table, and the table contents, an indirect call can be made to any memory location. This is illustrated diagrammatically in Fig. 6. All of the devices in the Z80 peripheral family are designed to operate in this mode of interrupt response. The programmer is able to specify a unique 8-bit interrupt vector to each peripheral, which it supplies to the c.p.u. during interrupt acknowledge. Interrupt priority is established by a "daisy-chain" connection through the peripheral devices.

References

1. The Zilog Z80/Z80A c.p.u. Technical Manual.

2. The Zilog Z80 Assembly Language Programming Manual.

Later this year we hope to publish a complete constructional design for a scientific computer using the Z80 m.p.u. as a processor.

WW diary overseas

The publishers of the Wireless World diary, T. J. & J. Smith of London SW19, do not supply direct to the public. If you want a copy you will have to get a bookseller, such as W. H. Smith, to order through the trade. If you live abroad from the UK, *Wireless World* can supply you. Send £1.50 to WW Diary, Room 25, Dorset House, Stamford Street, London SE1 9LU. The latest edition includes new sections on standard frequency transmissions, time code transmissions, UK broadcasting stations, and enlarges the address and telephone number section for electronics organisations by 75%.

Microelectronics design

Designing with single-chip microcomputers is the subject of one of the papers to be presented at the Microsystems '79 conference and exhibition this year, January 31 to February 2. Other topics covered are bubble memories, microprocessor interfacing, architecture of 16-bit processors, high level languages and costing m.p.u. software. The event will be at the West Centre Hotel, Lille Road, London SW6 from 09.30 to 18.00 hours each day. Conference details from IPC Science and Technology Press Ltd, Westbury House, Bury Street, Guildford, Surrey GU2 5AW (Tel: 0483 31261). Exhibition details from Iliffe Promotions Ltd, Dorset House, Stamford Street, London SE1 9LU (tel: 01-261 8000).