16 bit Microprocessor Architecture
Terry Dollhoff - 1979

5
F8

The F8 is a Fairchild invention and started out as a two chip microprocessor. Like the other 8-bit processors, the F8 has evolved into a whole cadre of machines. Some are single chip processors with program and data storage. The capability of each member of the F8 family is summarized in Figure 5-1. Most versions of the F8 are single chip processors which contain ROM and I/O.

Unlike the microprocessors we have discussed so far, it is necessary to explore more of the hardware aspects of the F8 to thoroughly understand its architecture. The original two chip F8 architecture is shown in Figure 5-2. The heart of any multi-chip F8 system is the 3850 processor. This chip provides the basic processing capability and working registers. It differs from other microprocessors because it does not include the program counter or memory address logic. They are contained in a companion chip. Removing the addressing logic from the CPU eliminates the need for an address bus which, in turn, reduces the external outputs of the chip. The CPU and support chips are all 40-pin chips. In addition to the ALU and working registers, two input/output ports, system clock generation, and power on reset are also included in the CPU.

In addition to the CPU chip, a multi-chip F8 system must include one or more of the following support chips:

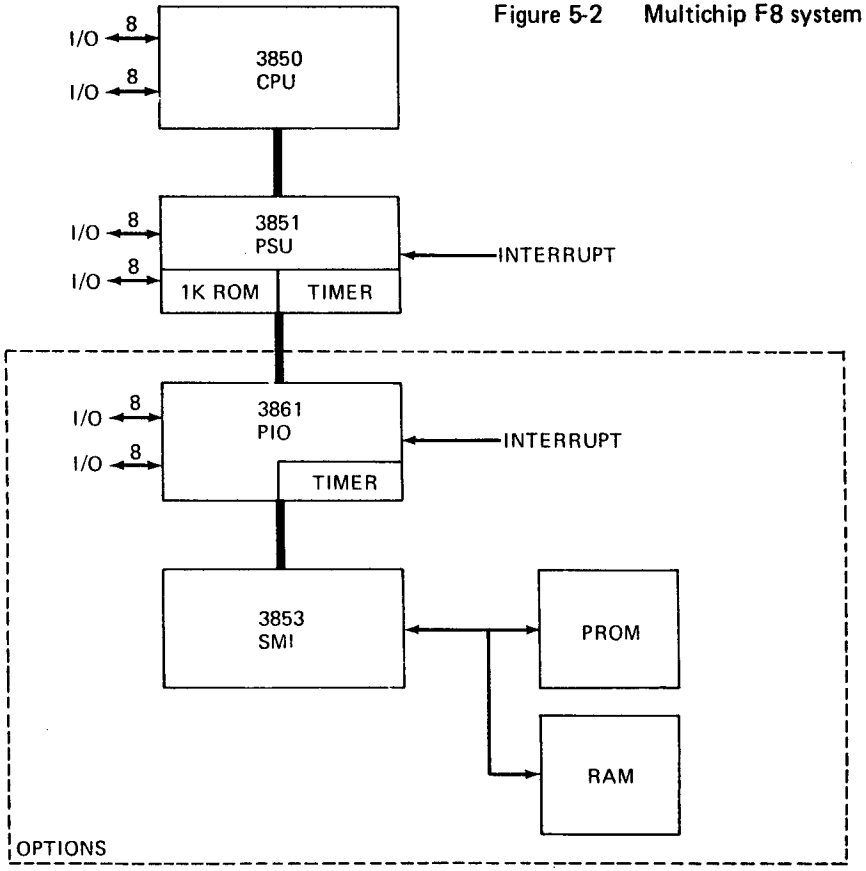
Program storage unit (3851) The program storage unit provides a masked ROM for program storage, four 8-bit input/output ports, a programmable internal timer, and external interrupt control. The

Figure 5-1 Single chip F8s

	On-chip ROM	On-chip RAM*	I/O
3870	2K ROM	64byte	4 ports
3872	4K ROM	128byte	4 ports
3876	2K ROM	128byte	4 ports

*Includes the 64-byte scratchpad RAM

Figure 5-2 Multichip F8 system



PSU in tandem with the CPU, creates a minimal two chip system. This combination provides 1K bytes of program storage (masked ROM), 8-bit input/output ports, 64 bytes of data storage or RAM, and a programmable internal timer. Even when it was first introduced, the cost of this system was low in comparison to other processors. This is probably why the F8 has always been a popular choice for equipment control.

Memory interface units (3853 and 3852) Since the CPU does not contain a program counter or other memory addressing registers, it is necessary to add another chip if the system must interface with external ROM, PROM, or RAM. The static memory interface (3853) is used to interface with static memories (RAM or ROM). The dynamic memory interface 3852 is used to interface with dynamic RAM. This chip provides the address and refresh circuits required by dynamic RAM. Both chips provide the address bus missing from the basic CPU chip.

Programmable I/O (3861) This chip is a subset of the PSU and includes all PSU features except masked ROM. It is intended to provide either additional I/O capability or another timer.

Direct memory access (3854) This chip allows an external device to access the memory. In some microprocessors, DMA is accomplished by forcing the processor to wait or hold. However, the F8 does not provide a hold or wait input. Therefore, it is necessary to add a DMA chip to the system if an external device must access memory. This does not represent a major increase in system components because other processors require extra external logic to control the hold input.

INTERNAL ARCHITECTURE

The F8 architecture is illustrated in Figure 5-3. It contains a single 8-bit accumulator and 64 scratchpad registers. One additional register, the indirect scratchpad address register (ISAR), is used to access many of the scratchpad registers because only a few of the scratchpad registers can be accessed directly. In fact, only the first 16 can be addressed by register addressing. The other 48 registers must be addressed via the ISAR.

In addition to the working registers, the F8 contains a 16-bit program counter (called P0), a data counter (DC), and program counter stack register (P). The program counter allows direct access to 64K of memory. The data counter register can be compared to the (H,L) register pair of the 8080 and is used to access external

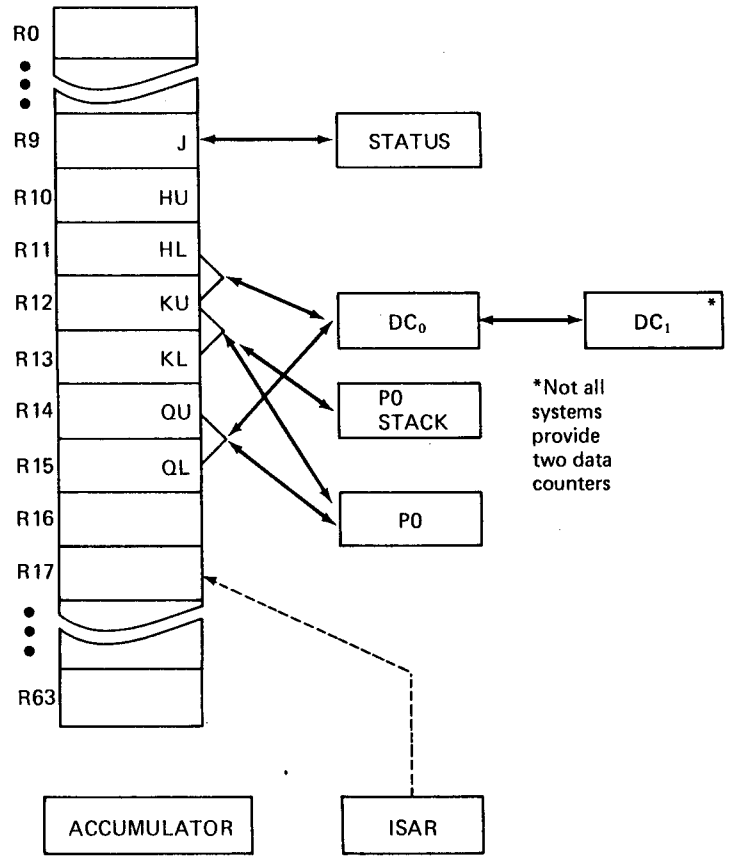


Figure 5-3 Register architecture

memory. This is the only vehicle for accessing operands from RAM in the F8. As we will see later, the program counter stack is used for subroutine and interrupt linkage, but it should not be confused with the stack pointer in other machines.

ADDRESSING

The F8 has a number of addressing modes, some of which we have not encountered in other processors. Register addressing is typical of register addressing in most machines. The operand is contained in one of the general registers. For arithmetic or logic operations, the first 12 general registers can be directly addressed as R0 to R11. For example:

```
AS  R6
```

will add the contents of R6 to the accumulator and place the result in the accumulator.

Scratchpad indirect addressing is like register indirect addressing, but the operand is contained in the scratchpad register (instead of memory) whose number is contained in ISAR. For example, if ISAR contains 31, then scratchpad register indirect selects scratchpad register 31. This is the only way to access registers 16 to 63. There are three variations of a scratchpad register indirect reference:

S the operand is selected by ISAR and ISAR remains unchanged.
I same as S, but the lower order 3 bits of ISAR are incremented after the scratchpad register indirect operation is performed.
D same as S, but the lower order 3 bits of ISAR are decremented after the scratchpad register indirect operation is performed.

Note that just the lower 3 bits of ISAR are incremented or decremented during a scratchpad autoincrement or decrement. This means that it is convenient to access scratchpad registers in groups of 8. Whenever the lower 3 bits of ISAR are all set to one, then a status register condition code indicates that fact. A branch on ISAR lower not equal to 7 (BR7) is part of the branch repertoire and is useful for loop control.

An immediate addressed operand is contained in the byte following the instruction. For example:

```
AI  6
```

will add 6 to the accumulator and store the result back in the accumulator. A variation of immediate addressing, immediate addressing/short form, is a unique mode of immediate addressing and can be used to decrease program size. The operand is contained in the lower 4 bits of the instruction byte itself. The following instruction will load the accumulator with the immediate value 4, but the instruction is only 1 byte long:

```
LIS  4
```

Relative addressing is used to determine the destination address for most jumps. The operand address is formed by adding the second byte of the instruction to the address of the instruction plus one. Notice that this calculation is different from the relative addressing calculation for the Z80 or 6800 because the offset is relative to the

second half of the current instruction instead of the next instruction. For example, the following instruction will transfer control to the address of the current instruction +6 (assume that the assembler enters the +5 directly into the instruction):

```
BR  +5
```

Direct addressing is available but it is restricted to just three instructions. The operand address is in the second and third bytes of the instruction. Direct addressing allows access to any of the 64K addressing space. The three instructions which allow direct addressing are:

Jump (JMP)
Call to subroutine (PI)
Load data counter (LR DC)

The F8 also provides register indirect addressing, but the operand address must be contained in the data counter register (DC0). After the operand has been referenced, the data counter is advanced by one. No other registers can be used for register indirect addressing.

USING THE F8

The F8 instruction set includes numerous side effects which are not typical and are worthy of special attention. If you are not aware of the oddities, you may incorporate bugs into your program which are difficult to locate. The status register condition codes differ from most other micros. The carry and zero flags are the same as usual, but the sign flag is different. In particular, S=1 implies that the result was plus, not minus. If you stick to the basic branches, it should not present a problem. If you prefer to "roll your own" using the branch on condition true or false, be careful not to reverse the intended test.

Whenever an extended jump or subroutine call is executed, the accumulator is modified. This is clearly stated in the manual, but is all too easy to forget. It also eliminates one method of passing parameters to the subroutines.

The odd thing about the F8 is that the system can have more than one PC or DC! Anytime the PC or DC registers are loaded, all PC or DC registers in the system are loaded. This multiplicity of PCs and DCs is transparent to the programmer (it simplifies the hardware by avoiding address transmissions between chips), or so it seems. One problem is caused because the F8 provides two data counters, DC0

and DC1, but some of the support chips in the system only provide one, DC0. Whenever the data counter is loaded, all DC0 registers are loaded accordingly. However, when the data counters are exchanged, the chips with one data counter simply ignore the exchange. If the system contains one chip with two data counters and one with only one, data counter exchanging can produce improper memory addressing.

The F8 performs compare operations differently from other microprocessors. In particular, the accumulator is subtracted from the compare value instead of vice versa. This is not an architectural shortcoming, but it certainly can be confusing. While we are discussing the compare, you might also note that there is no way to compare the accumulator with the scratchpad registers. The accumulator can only be compared with a constant or memory.

There are several alternatives to comparing the accumulator with the scratchpad. One is to place the scratchpad contents in RAM and then compare the accumulator with memory. This is practical due to the RAM addressing scheme. DC0 is the only way to address RAM, and it is incremented after each reference. Another way to compare the accumulator with scratchpad values is to use the logical or arithmetic instructions. To compare for equality, just exclusive-OR the scratchpad with the accumulator.

The F8 has decimal addition and subtraction instructions to simplify handling BCD data. They must be used cautiously. Prior to a decimal add, the constant 66 (hexadecimal) must be added to the accumulator. Then the decimal add can be done.

The F8 has one other idiosyncrasy. It does not provide a two's complement instruction nor does it provide a subtract. Therefore, a two's complement subtraction requires three instructions and it is still done in reverse of normal ordering:

COM	1's complement of A
INC	increment A (now two's complement)
AS Ri	add register to A

SUBROUTINE INTERFACE

When a subroutine call is executed, the F8 places the return address in the program counter stack register, P. If the called routine makes further calls, this register must be preserved. Even if the system can get by with one level of subroutine call, it may still be necessary to save this register because an interrupt sequence is performed

the same as a subroutine call. Therefore, if the subroutine does not immediately save the program counter stack and an interrupt is honored, the return address is lost. The return address can be stored in a scratchpad register.

INTERRUPTS

As already mentioned, the F8 treats an interrupt just like a subroutine call. The return address is stacked in the program counter stack register, P, and control is transferred to the interrupt address. There is a significant fact, carefully hidden in most F8 literature, that may impact software design; namely, external interrupts and timer interrupts from the same chip cannot be enabled at the same time. This makes it difficult, if not impossible, to use both features simultaneously.

INSTRUCTION SET

The following summarizes all of the instructions for the F8. The table includes the execution time in short cycles for each instruction. A short cycle is four times the basic clock rate. For example, if the clock rate is 2MHz (.5 microseconds) then a short cycle is 2.0 microseconds. If two times are shown (e.g., 3/3.5) the longest time is for a jump that is taken. In the summary, the following notations are used:

A accumulator
Ri scratchpad register 'i' (e.g., R11)
P0 program counter
P program counter stack
DC0 memory address register
DC1 alternate memory address register
W status register
t 3-bit status selection
ISAR scratchpad address register. This register is used to address scratchpad registers via register indirect addressing.
r scratchpad addressing as:
- **0 to 11** select R0 to R11
- **I** select (ISAR), then ISAR=ISAR+1
- **S** select (ISAR)

D select (ISAR), then ISAR=ISAR-1
i 4-bit constant
n 8-bit constant
nn 16-bit constant
( ) contents of memory (e.g., (DC))

F8

Mnemonic	Len	Cycles	Description
8-bit transfers			
LR A,r	1	1	A=r
LR A,Ku	1	1	A=R12
LR A,KI	1	1	A=R13
LR A,Qu	1	1	A=R14
LR A,QI	1	1	A=R15
LR r,A	1	1	r=A
LR Ku,A	1	1	R12=A
LR KI,A	1	1	R13=A
LR Qu,A	1	1	R14=A
LR QI,A	1	1	R15=A
LM	1	2.5	A=(DC0) DC0=DC0+1
ST	1	2.5	(DC0)=A DC0=DC0+1
LR A, IS	1	1	A=ISAR
LR IS,A	1	1	ISAR=A
LR J,W	1	1	R9=W
LR W,J	1	2	W=R9
LISL i	1	1	ISAR(lower)=i
LISU i	1	1	ISAR(upper)=i
LI n	2	2.5	A=n
LIS i	1	1	A=i
CLR	1	1	A=0
16-bit transfers/arithmetic			
LR K,P	1	4	R12=P(upper), R13=P(lower)
LR H,DC	1	4	R10=DC0(upper), R11=DC0(lower)
LR Q,DC	1	4	R14=DC0(upper), R15=DC0(lower)
LR P,K	1	4	P(upper)=R12, P(lower)=R13
LR DC,H	1	4	DC0(upper)=R10, DC0(lower)=R11
LR DC,Q	1	4	DC0(upper)=R14, DC0(lower)=R15
DCI nn	3	6	DC0=nn
XDC	1	2	DC0=DC1, DC1=DC0
ADC	1	2.5	DC0=DC0+A
8-bit arithmetic			
AS r	1	1	A=A+r
AM	1	2.5	A=A+(DC0), DC0=DC0+1
AI n	2	2.5	A=A+n
ASD r	1	2	A=A+r (decimal adjusted)
AMD	1	2.5	A=A+(DC0) (decimal adjusted) DC0=DC0+1
NS r	1	1	A=A AND r
NM	1	2.5	A=A AND (DC0), DC0=DC0+1
NI n	2	2.5	A=A AND n

Mnemonic	Len	Cycles	Description
XS r	1	1	A=A XOR r
XM	1	2.5	A=A XOR (DC0), DC0=DC0+1
XI n	2	2.5	A=A XOR n
OM	1	2.5	A=A OR (DC0), DC0=DC0+1
OI n	2	2.5	A=A OR n
CM	1	2.5	compare (DC0) to A DC0=DC0+1
CI n	2	2.5	compare n to A
INC	1	1	A=A+1
DS r	1	1.5	r=r-1
LNK	1	1	A=A+CY
Jumps			
BR n	2	3.5	P0=P0+1+n
BC n	2	3/3.5	IF CARRY THEN P0=P0+1+n
BNC n	2	3/3.5	IF NO CARRY THEN P0=P0+1+n
BP n	2	3/3.5	IF POSITIVE THEN P0=P0+1+n
BM n	2	3/3.5	IF NEGATIVE THEN P0=P0+1+n
BZ n	2	3/3.5	IF ZERO THEN P0=P0+1+n
BNZ n	2	3/3.5	IF NOT ZERO THEN P0=P0+1+n
BNO n	2	3/3.5	IF NO OVERFLOW THEN P0=P0+1+n
BT t,n	2	3/3.5	IF t THEN P0=P0+1+n
BF t,n	2	3/3.5	IF NOT t THEN P0=P0+1+n
BR7 n	2	2/2.5	IF ISAR(lower) NOT 7 THEN P0=P0+1+n
JMP nn	3	5.5	P0=nn, A is destroyed
LR P0,Q	1	4	P0(upper)=R14, P0(lower)=R15
Call/return			
PK	1	2.5	(call to 'K') P=P0 P0(upper)=R12, P0(lower)=R13
PI nn	3	6.5	(call direct) P=P0 P0=nn A is destroyed
POP	1	2	(return) P0=P
Shifts			
SL 1	1	1	shift A left one, zero fill
SL 4	1	1	shift A left four, zero fill
SR 1	1	1	shift A right one, zero fill
SR 4	1	1	shift A right four, zero fill
Misc.			
COM	1	1	A=1's complement of A
NOP	1	1	no operation
EI	1	1	enable interrupts
DI	1	1	disable interrupts
Input/output			
IN n	2	4	input (n) to A
INS i	1	2(i=0,1) 4(i=4,5,6,7)	input (i) to A
OUT i	2	4	output (n) from A
OUTS i	1	2(i=0,1) 4(i=4,5,6,7)	output (i) from A