A Powerful Microprogram Control Unit The 6710

Monolithic Memories, Inc.
1165 East Arques Avenue
Sunnyvale, California 94086
Tel (408) 739-3535 TWX 910—339-9229

A Powerful Microprogram Control Unit — The 6710

by
Clive Ghest
Monolithic Memories, Inc.
Sunnyvale, California

INTRODUCTION

The Monolithic Memories 6710 (MCU) is a powerful Microprogram Control Unit or Sequencer which can be used either as the control element in a system incorporating the MMI 6701 Microcontroller, or as a stand-alone sequence controller in non-arithmetic applications.

The MCU is used with standard Random Access Memory RAM, ROM, or PROM, with a portion of the memory output providing information required by the MCU and the remainder used to provide the signals necessary for the 6701 and peripheral logic circuits. The MCU can address up to 512 words of control memory directly. (This can be expanded by paging techniques.) The MCU has an extremely powerful but straightforward instruction set which includes conditional jumps, subroutine jumps, etc. The MCU also has the capability of multi-way branching at each microstep enabling complex operations such as multiply, divide, etc. to be perforemd in a minimum number of control memory words and be performed at high speed. The MCU also has, in addition to a single level of subroutine, an on-chip control counter, thereby allowing repetitive microsteps to be easily programmed without the necessity of externally gating the clock.

The MCU is designed so that when used in conjunction with the 6701 Microcontroller it provides all the necessary control and accepts all the status conditions required in minimum but powerful systems. Provision is made for conditional branching on Carry, Overflow, Zero, etc., and the control signals are provided for a small but adequate set of shifting options.

The MCU requires an 18 bit field for its instructions. It is packaged in a 40 pin DIP Package and operates at a clock rate of greater than 10 MHz. In a practical system without pipelining of instructions the microstep time should be typically less than 250 ns using a control memory with an access time of 50 ns.

TYPICAL SYSTEM CONFIGURATION

The block diagram of a typical system using the MCU and the 6701 is shown in Figure 1. The control memory address register of the MCU addresses 512 word pages of control memory. This memory output is divided up into several fields. One field of up to 18 bits provides the signals needed by the MCU for generation of the next memory address and thereby the sequence of programmed microsteps of the system instructions. The second field provides the 6701 Microcontroller the instruction control required by the present microstep. The third field is required to control external memory, peripheral equipments, interrupts, DMA, etc. The various output fields from the control memory are shown for a typical system in Figure 2.

The clocks of the MCU and Microcontroller are connected so that all operations are synchronous and no critical races are present. The microcycle time is made up of the propagation delay from the clock edge to the address output of the MCU, the delay through the control memory, and the set up time required for correct operation of the 6701. If, as is likely, the system accepts instructions from the main memory and the system is programmed at the instruction level, the first action necessary after obtaining the instruction from memory is to indicate to the MCU the starting address of the instruction. The instruction portion of the instruction register is forced onto the MCU address inputs, so that the MCU can jump to the start of the microinstruction sequence defined by the operation code in the instruction register. Since the control memory normally supplies the jump information, the instruction register operation field and output of the control memory must be multiplexed. In practice, a bus oriented scheme is advantageous and the instruction register and this control memory portion can have open collector or three state outputs. The active element can then be defined by a micro-bit in the control word, or by the system recognizing the beginning of an instruction. After the microinstruction sequence is initiated the MCU continues through the microsteps programmed in the control memory which make up the instruction.

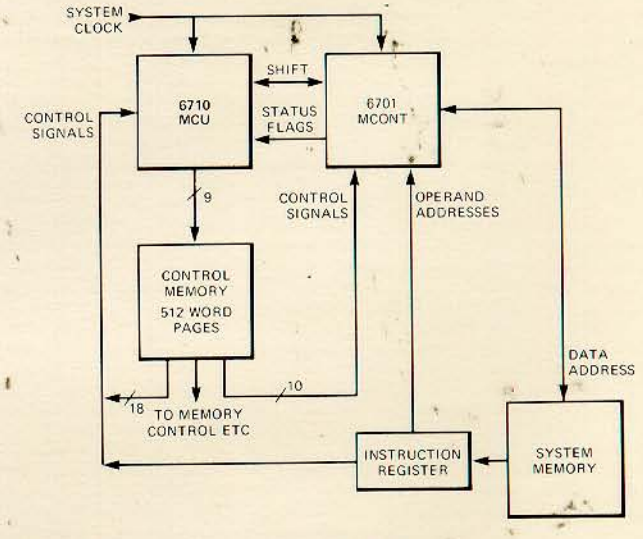


FIGURE 1. Typical System Block Diagram

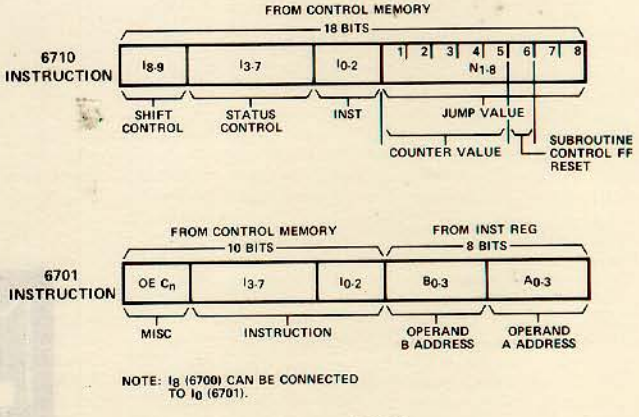


FIGURE 2. System Instruction Fields

MCU ORGANIZATION

The basic organization of the MCU is shown in Figure 3. It consists of three main sections: an Internal Control section which has a 9 bit CRAR (Control ROM Address Register) with its subroutine storage register, and a 5 bit

CC (Control Counter) with its subroutine storage register, the Shift Logic section with provision for a minimum set of shifting operations, and the third section, the Flag Status Logic for providing the multiplexing and storage of the various Flag signals.

INTERNAL CONTROL

The 9 bit Control ROM Address Register (CRAR) is divided up into two parts. The most significant 8 bits are completely separate from the least significant bit. The 8 bit section can be incremented, incremented and stored in a Subroutine Return Register, or can be set from the Control Data Inputs. This 8 bit section allows every other CROM Address to be called up in sequence, allows unconditional Jumping or Subroutine Jumping. The 1 bit section allows every microcycle to be a conditional Branch to an odd or even CROM address.

The internal working of the MCU is controlled by a 3 bit MCU instruction Control Field I_{0-2}. As shown in Table 1, this 3 bit field allows the MCU to perform 8 different types of operation, such as Continue, Jump, Conditional Jump, Subroutine Jump, Conditional Subroutine Jump, etc. An internal control counter of 5 bits allows a sequence of instructions to be repeated up to 32 times enabling the MCU to be used efficiently with the machines having up to 32 bit word lengths. The counter can be loaded from the input data bus N_{1-5} and is decremented on conditional Jump instructions and tested for zero. During conditional Jump operations if the counter is zero the MCU ignores the Jump and continues on to the next microinstruction in sequence. This Conditional Jump feature can be used also with Subroutines, thereby allowing a Subroutine to be called up a specific number of times. The Control Counter is stored during Subroutines in a temporary storage register so the Control Counter can be used during subroutines. The temporary storage register is also used in the preprogrammed subroutine re-entry mode which is initialized by the control code $I_0I_1I_2 = 101$.

The instruction code 101 signifies a Subroutine Jump with the return not stored in the control memory, but preprogrammed in the control counter. The machine Jumps to a Subroutine and continues through a sequence until the value of the CRAR least significant 5 bits is the same as the control counter, whereupon the MCU automatically returns to the calling program. This means that instructions can be built up from small sections of existing microcode sequences. This feature is very valuable when additional instructions are to be added, or an error is discovered in the code. The main bulk of the code need not be disturbed; patches to the code do not cause a complete change of control memory.

The four way branches are performed by incorporating a conditional branch with a conditional Jump. The 5 bit Control Counter can be loaded from the N_{1-5} data inputs during $I_0I_1I_2$ = 100 instruction. Whenever a 010 Conditional Jump or 110 Conditional Subroutine Jump is encountered the Control Counter is checked to see if it is zero and decremented. If the counter is not zero then a Jump occurs to the address N_{1-8} and the value of the Conditional status output line. If the counter is zero then the MCU continues on to the next pair of addresses as defined by the Conditional status output. This conditional Jump capability allows a section of code to be repeated N or N+1 times or allows a Subroutine to be repeated N or N+1 times.

TABLE I. MCU Control Options

CONTROL CODE I_2	I_1	I_0	ADDRESS FIELD DESTINATION	CONTROL ACTION
0	0	0	NONE	CONTINUE TO NEXT μINSTRUCTION
0	0	1	CONTROL COUNTER	CONTINUE TO NEXT μINSTRUCTION
0	1	0	NONE/CRAR (COND. JUMP)	JUMP TO NEXT μINSTRUCTION IF CONTROL COUNTER ≠ 0, DECREMENT CONTROL COUNTER
0	1	1	NONE/CRAR (COND. SUBR. JUMP)	SUBROUTINE JUMP TO NEXT μINSTRUCTION IF CONTROL COUNTER ≠ 0, DECREMENT CONTROL COUNTER
1*	0	0	NONE	RETURN FROM SUBROUTINE
1**	0	1	CRAR (JUMP SUBROUTINE)	JUMP TO NEXT μINSTRUCTION RETURN FROM SUBROUTINE WHEN SUBROUTINE CONTROL COUNTER LATCH = $CRAR_{0-4}$
1	1	0	CRAR (JUMP)	JUMP TO NEXT μINSTRUCTION
1	1	1	CRAR (JUMP SUBROUTINE)	SUBROUTINE JUMP TO NEXT μINSTRUCTION

* The machine will only return to the calling program if it entered a subroutine via a subroutine jump instruction otherwise it will continue to the next μinstruction in sequence.

** This operation allows the controller to branch to a section of code, perform the operations outlined by the code and return after a preprogrammed CROM address has been reached or if a return is encountered.

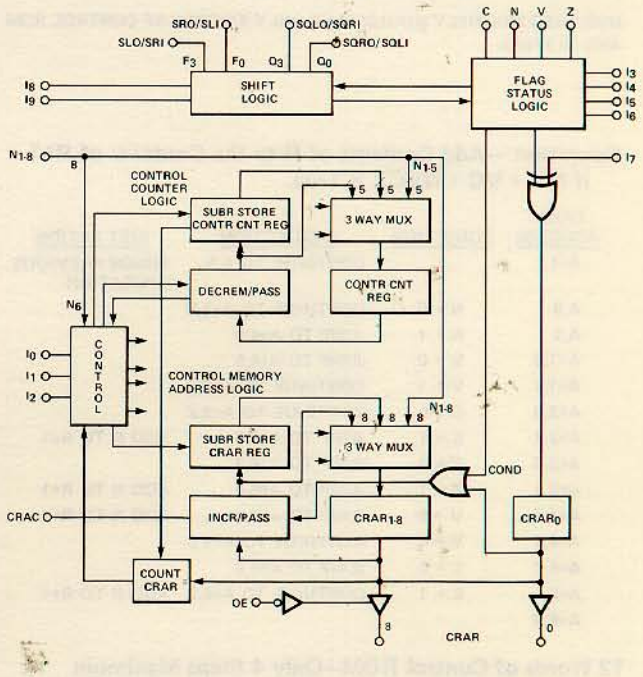


FIGURE 3. MCU Block Diagram

SHIFT CONTROL

Since pins are at a premium the approach taken has been to provide in the MCU the basic minimum shifting options, shown in Figure 4 and Table II, for a system using the 6701 Microcontroller. The 6701 provides on two bidirectional buses, SLO/SRI (Shift Left Out/Shift Right In) and SRO/SLI (Shift Right Out/Shift Left In), access to the least or most significant bit of the result of an operation F_0 and F_3 respectively. The 6701 also has an Accumulator or double length extension register Q which can be shifted in conjunction with the operation result. Again access to the least or most significant bit of the Q register Q_0 or Q_3 is available via the bidirectional buses SQLO/SQRI and SQRO/SQLI.

The MCU provides the signals to these four bidirectional buses to derive a small but powerful set of shifting variations. The MCU provides signals for double length operations, which also satisfies the majority of single length shifting requirements. The only minor drawback is that during a single length Arithmetic shift left Q_3 is loaded into the selected register least significant bit.

During a double length End Around Left Shift, or an Arithmetic Left Shift, the Flag Register is shifted from the Flag Status Logic. If the Flag logic shift is allowed by the Flag Status code the complete 4 bit word Flag Register C, N, V, Z is shifted. This mode is used for the storage and retrieval of the Flag Register during interrupts. However, if the Flag Register is not allowed to shift, C is inserted into Q but the Flags maintain their position in the Flag Register. Obtaining access to C in this manner allows C to enter the Q register without disturbing the Flag positions; this is very useful during a division operation.

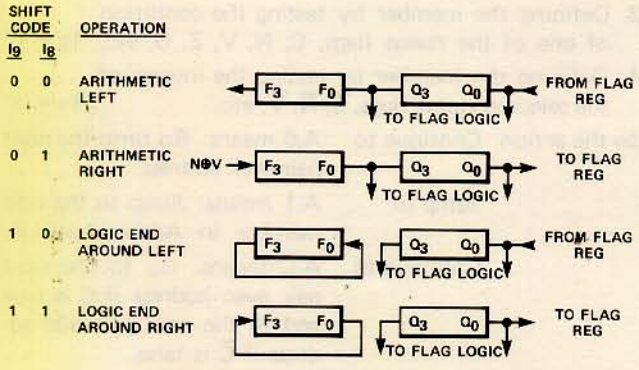


FIGURE 4. Shift Options

TABLE II.

CONTROL CODE		SHIFTING OPERATION	SLO/SRI	SRO/SLI	SQLO/SQRI	SQRO/SQLI
I_9	I_8		(F_3)	(F_0)	(Q_3)	(Q_0)
0	0	ARITH. LEFT SHIFT	–	Q_3	–	FLAG (SC)
0	1	ARITH. RIGHT SHIFT	N⊕V	–	F_0	–
1	0	ROTATE LEFT	–	F_3	–	FLAG (SC)
1	1	ROTATE RIGHT	F_0	–	–	–

– HIGH IMPEDANCE
SC CARRY FLIP FLOP

FLAG CONDITIONAL STATUS CONTROL

Because the power of the MCU is based to a large extent on the flag conditional status logic and the ability of the machine to perform operations dependent on many flag conditions, the choice has been made to make this section of the MCU extremely powerful. Four flags – C (Carry), N (Negative), V (Overflow), and Z (Zero) – can be stored, and these stored values and the instantaneous values are available for two way decision branching. In addition, the Q_0, Q_3 outputs are also available for making a branch decision. Access to Q_0 is required for multiplication routines, and to Q_3 for justification during double length working. Any condition can be inverted, so that the branching decision can be made on the inverse of C, N, etc. This feature is very useful in Exclusive OR and Parity Generation. Figure 5 shows the block diagram of the Flag Status Control Logic, and the various Flag Status Control options are shown in Table III.

If additional conditional Flag inputs are required an external multiplexer can be used with the select field of the multiplexer being driven from additional control memory inputs.

The Flag Status Control Code also includes the ability to shift the Flag register to and from Q so that the register can be stored during interrupts and also assist in some complex shifting instructions.

TABLE III. Flag Status Control Options

CONTROL CODE					ACTION	
I_6	I_5	I_4	I_3			
0	0	0	0	NONE	I_7 TO $CRAR_0$	UNCONDITIONAL BRANCH
0	0	0	1	STORE C	I_7 TO $CRAR_0$	UNCONDITIONAL BRANCH
0	0	1	0	STORE N, V, Z	I_7 TO $CRAR_0$	UNCONDITIONAL BRANCH
0	0	1	1	STORE C, N, V, Z	I_7 TO $CRAR_0$	UNCONDITIONAL BRANCH
0	1	0	0	SHIFT FLAG REGISTER INTO Q_0	I_7 TO $CRAR_0$	UNCONDITIONAL BRANCH
0	1	0	1	SHIFT FLAG REGISTER OUT OF Q_0	I_7 TO $CRAR_0$	UNCONDITIONAL BRANCH
0	1	1	0	INSTANTANEOUS VALUE OF Q_3 TO $CRAR_0$		CONDITIONAL BRANCH
0	1	1	1	INSTANTANEOUS VALUE OF Q_0 TO $CRAR_0$		CONDITIONAL BRANCH
1	0	0	0	STORED VALUE OF C TO $CRAR_0$		CONDITIONAL BRANCH
1	0	0	1	STORED VALUE OF N TO $CRAR_0$		CONDITIONAL BRANCH
1	0	1	0	STORED VALUE OF V TO $CRAR_0$		CONDITIONAL BRANCH
1	0	1	1	STORED VALUE OF Z TO $CRAR_0$		CONDITIONAL BRANCH
1	1	0	0	INSTANTANEOUS VALUE OF C TO $CRAR_0$		CONDITIONAL BRANCH
1	1	0	1	INSTANTANEOUS VALUE OF N TO $CRAR_0$		CONDITIONAL BRANCH
1	1	1	0	INSTANTANEOUS VALUE OF V TO $CRAR_0$		CONDITIONAL BRANCH
1	1	1	1	INSTANTANEOUS VALUE OF Z TO $CRAR_0$		CONDITIONAL BRANCH

Code Bit I_7 inverts the status of output line so that the condition is dependent upon $\overline{C}$, etc. For the first six entries in the table if $I_7 = 0$ there is an unconditional branch to X,0. If $I_7 = 1$ an unconditional branch to X, 1.

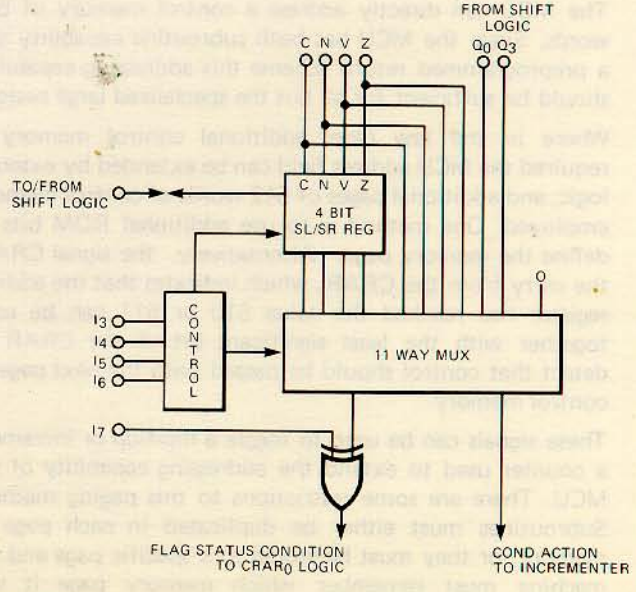


FIGURE 5. Flag Status Block Diagram

INTERRUPTS

Interrupts must only be accepted at the end of an instruction. The carry signal CRAC from the address register can be used to indicate to external interrupt logic that the end of an instruction has been reached and interrupts are allowed. External logic can then, if an interrupt is waiting, float the output of the CRAR and force the CROM to a certain address where the interrupt subroutine begins. Since an 8 bit field is available this allows any number of levels of interrupt in a system. Alternatively, the output of the CROM can be inhibited and the Subroutine Jump instruction together with the Jump Address can be forced onto the CROM output bus.

The next problem with interrupts is that not only do the internal working registers have to be stored if they are used during the interrupt subroutine, but also any flags which are in the Flag register and might be scanned after the interrupt. The method chosen, since pins are scarce, is to shift the status register into the Q register of the 6701 or an external register hanging on to the Q shift pin. After the subroutine the Q register can be shifted back into the status register. Both of these operations take 4 microcycles to store and replace the Flags. Once the Flags are in the Q register they can then be stored in the working registers of the 6701, or they can be written into the main storage so that multiple interrupt levels are possible.

Microinterrupts from peripheral devices may need to stop the normal sequence of microinstructions at any step in any instruction. This can be done in a similar manner to interrupts, although the MCU does not have to indicate that the end of an instruction has been reached. Whether the flags have to be stored depends upon the specific requirements of the system.

The DMA function can be derived by causing an interrupt at the end of an external memory cycle. The signals controlling the external memory from the CROM can be used to indicate this condition. During a DMA operation the MCU flags would not generally be stored.

CONTROL MEMORY EXPANSION

The MCU can directly address a control memory of 512 words. Since the MCU has both subroutine capability and a preprogrammed return scheme this addressing capability should be sufficient for all but the specialized large system.

Where in the few cases additional control memory is required the MCU address field can be extended by external logic, and additional pages of 512 words of control memory employed. One method is to use additional ROM bits to define the memory page. Alternatively, the signal CRAC, the carry from the CRAR, which indicates that the address register has reached the value 510 or 511 can be used together with the least significant bit of the CRAR to detect that control should be passed onto the next page of control memory.

These signals can be used to toggle a flip-flop or increment a counter used to extend the addressing capability of the MCU. There are some restrictions to this paging method. Subroutines must either be duplicated in each page of memory, or they must be stored in a specific page and the machine must remember which memory page it was working in prior to the subroutine.

PROGRAMMING THE MCU

The basic organization and philosophy of the MCU in having a combinatorial branch at each microcycle not only makes for an extremely powerful machine but also simplifies the programming enormously. This simplification should be a key feature of the MCU. Rather than think in terms of conditional jump methods to track through a flow chart, the flow chart can be essentially duplicated in the hardware. This should make the programming task faster, easier to understand, less prone to error, and allow easier changes.

The key to understanding the working of the MCU is to divide up the Addresses in the Control ROM into pairs, each with an odd and even address. The MCU then has the capability of defining the next address by calling up the Address pair, say A, and the even or odd address in the pair by one of four ways. These ways are:

1. Defining the even member of the pair — I_7 = '0'
2. Defining the odd member of the pair — I_7 = '1'
3. Defining the member by testing the condition of one of the status flags, C, N, V, Z, Q, etc. — I_7 = '0'
4. Defining the member by testing the inverse of the selected status flags, $\overline{C}$, $\overline{N}$, $\overline{V}$, etc. — I_7 = '1'

So the action

Continue to	A,0 means: Go onto the next pair even address.
Jump to	A,1 means: Jump to the odd member in Address pair A.
Continue to	A,$\overline{C}$ means: Go to the next pair even address if C is true and to the next pair odd address if $\overline{C}$ is false.

Since the addresses are carried in pairs, the targets of a conditional branch must always start at an even address. The conditional pair cannot cover the addresses (A,1) (A+1,0). This is not true for unconditional addresses which can step from odd to even to odd and so on. Because conditional branches are in pairs, occasionally, an odd address is skipped. These addresses are not lost and can still be used by jumping to and from them.

In conclusion we give some examples of programming the MCU to solve complex arithmetic and logic problems.

Operation —Add Contents of R to the Contents of R+1 if N⊕V is True.

CROM ADDRESS	CONDITION	MCU ACTION	6701 ACTION
A-1,1		CONTINUE TO A,N	FINISH PREVIOUS OPERATION
A,0	N = 0	CONTINUE TO A+1,V	
A,1	N = 1	CONTINUE TO A+1,$\overline{V}$	
A+1,0	V = 0	JUMP TO A+2,0	
A+1,1	V = 1	CONTINUE TO A+2,0	ADD R TO R+1

4 Words of Control ROM—2 Steps

NOTE AN N VARIABLE EXCLUSIVE OR TAKES ONLY 2N WORDS OF CONTROL ROM AND N STEPS. EVERY 2 VARIABLE FUNCTION CAN BE PERFORMED WITH JUST 4 WORDS AND 2 STEPS.

Operation —16-bit 2s Complement Multiply

MULTIPLY THE CONTENTS OF R BY THE CONTENTS OF R+1 TO FORM A DOUBLE LENGTH PRODUCT WITH THE MOST SIGNIFICANT HALF IN R+1 AND THE LEAST SIGNIFICANT IN R.

CROM ADDRESS	CONDITION	MCU ACTION	6701 ACTION
A-1,1		CONTINUE TO A,0	FINISH PREVIOUS OPERATION
A,0		CONTINUE TO A,1	R+1 TO Q
A,1		SET CONTR. CNTR. TO 14 CONTINUE TO A+1,Q_0	CLEAR R+1, SHIFT ARITH RIGHT R+1,Q
A+1,0	$Q_0 = 0$	JUMP TO A+1,Q_0 IF CC≠0 DECREMENT CC.	SHIFT ARITH RIGHT R+1,Q
A+1,1	$Q_0 = 1$	JUMP TO A+1,Q_0 IF CC≠0 DECREMENT CC.	ADD R TO R+1, SHIFT ARITH RIGHT RESULT,Q
A+2,0	$Q_0 = 0$	JUMP TO A+3,0	SHIFT ARITH RIGHT R+1,Q
A+2,1	$Q_0 = 1$	JUMP TO A+3,0	SUBTRACT R FROM R+1 SHIFT ARITH RIGHT RESULT,Q
A+3,0		CONTINUE TO A+3,1	Q TO R

7 Words of Control ROM—19 Steps

UNSIGNED MULTIPLY WOULD TAKE ONLY 5 WORDS OF CONTROL ROM AND 19 STEPS.

Operation —Add Contents of R to the Contents of R+1 if $N\overline{V} + VC + \overline{N}V\overline{C}Z$ is True.

CROM ADDRESS	CONDITION	MCU ACTION	6701 ACTION
A-1,1		CONTINUE TO A,N	FINISH PREVIOUS OPERATION
A,0	N = 0	CONTINUE TO A+1,V	
A,1	N = 1	JUMP TO A+4,V	
A+1,0	V = 0	JUMP TO A+6,0	
A+1,1	V = 1	CONTINUE TO A+2,C	
A+2,0	C = 0	CONTINUE TO A+3,Z	
A+2,1	C = 1	JUMP TO A+6,0	ADD R TO R+1
A+3,0	Z = 0	JUMP TO A+6,0	
A+3,1	Z = 1	JUMP TO A+6,0	ADD R TO R+1
A+4,0	V = 0	JUMP TO A+6,0	ADD R TO R+1
A+4,1	V = 1	CONTINUE TO A+5,C	
A+5,0	C = 0	JUMP TO A+6,0	
A+5,1	C = 1	CONTINUE TO A+6,0	ADD R TO R+1
A+6,0			

12 Words of Control ROM—Only 4 Steps Maximum

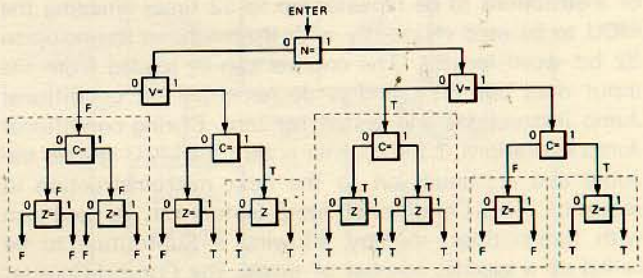
Same problem but with minimization:

CROM ADDRESS	CONDITION	MCU ACTION	6701 ACTION
A-1,1		CONTINUE TO A,V	FINISH PREVIOUS OPERATION
A,0	V = 0	CONTINUE TO A+1,N	
A,1	V = 1	JUMP TO A+2,C	
A+1,0	N = 0	JUMP TO A+4,0	
A+1,1	N = 1	JUMP TO A+4,0	ADD R TO R+1
A+2,0	C = 0	CONTINUE TO A+3,Z	
A+2,1	C = 1	JUMP TO A+4,0	ADD R TO R+1
A+3,0	Z = 0	JUMP TO A+4,0	
A+3,1	Z = 1	JUMP TO A+1,$\overline{N}$	
A+4,0			

8 Words of Control ROM—Only 4 Steps Maximum

EXAMPLE: If $N\overline{V} + VC + \overline{N}V\overline{C}Z$ is true (T) add C (R) to C (R + 1), if false (F) do nothing.

Straightforward solution every condition inside dotted area can be removed.



6 Blocks to give 12 Words CROM—4 Steps Maximum

A Minimum Solution