AccessionIndex: TCD-SCSS-T.20250922.003
Accession Date: 22-Sep-2025
Accession By: Dr.Brian Coghlan
Object name: Monolithic Memories 6700 bit-slice chipset
Vintage: 1974
Synopsis: A bipolar microprogrammed bit-slice processor.

Description:
Monolithic Memories [1][2] introduced its 5700/6700 bipolar bit-slice chipset in 1974, ultimately consisting of a microprogram control unit and an arithmetic unit, both intended for use in microprogrammed computer architectures.

Microprogramming was invented by Maurice Wilkes at Cambridge University, U.K. in 1951 [3]. Even by then a computer was understood to consist of a central processing unit (CPU), memory and input/output. Wilkes proposed that the CPU hardware would be simpler if a very basic CPU could be *microprogrammed* to emulate a higher-level instruction set of the user-level CPU and thereby execute programs in the user-level memory (Fig.1). This simple CPU would be composed of a microprogram control unit, microprogram memory and a central processing element.

This concept was of great value while hardware (both logic and memory) was expensive, as it was for the vacuum-tube era of the early 1950s and the transistor era of the late 1950s and early 1960s, as well as for the early microprocessor era of the 1970s and 1980s. Since then the ability to fabricate billions of inexpensive transistors on a chip has rendered this a niche value concept.

MMI designed the 6700 chipset in the mid-1970s to implement this concept. The microprogrammed CPU would comprise a 67110 microprogram sequencer, microprogram memory and multiple 6701 arithmetic units. The sequencer would address its own microprogram memory, with some of its contents being used to control the program sequence. But other of the contents would be distributed to the arithmetic units to control their operations. Adding a pipeline register would speed up execution.

The 5700 and 6700 chips were initially introduced in 1974 as a pair of 4-bit bit-slice microcontrollers, marketed to replace what would otherwise be around 25-30 discrete logic chips. No supporting chips or design aids of any kind were offered to make their use any easier. Typically multiple of the chips were chained together to create larger word sizes. The 5701 was the ‘military’ family rated at a clock frequency of 4MHz, whilst the 6701 was the ‘commercial’ family rated at 5MHz (later versions were rated up to 11MHz). They were similar to Texas Instrument’s SN74181, but with the addition of a register file and support logic to make it more adaptable, integrating about 1000 gates, see the chip die micrograph in Fig.2. The Soviets based their 8-bit 1802VS1 on it. The crucial supporting 67110 microprogram controller was only announced two years later in 1976, see Fig.3.

The 5701/6701 were vertically complete yet horizontally expandable 4-bit processing elements. With the contemporary LSI bipolar technology, these 4-bit slices were the largest that could be economically produced. They reduced the required components by half compared to the 2-bit Intel 3000 family (see elsewhere in this catalog). Fig.4

depicts the internal architecture of the 6701, and Fig.5 shows its pinout. They included full internal carry lookahead, shifting, true/complement operands, an auxiliary 'Q' register for multiply and divide implementation, and negative, positive, zero & overflow detection. Like the later AMD Am2901 (again see elsewhere in this catalog) they incorporated a 16-word by 4-bit 2-port register file. This had two independent read addresses A and B, where B was also the write address. There were no restrictions on addressing the register file. A 'Q' Register could function as an extension to the accumulator or as an additional register with full shifting capabilities. Normally, this 'Q' Register was used to implement multiplication and division. Complete single and double length arithmetic and logical shifting could be performed with various end condition propagation and/or deletions. Fig.6 shows the 5701/6701 detailed internal block diagram, and Fig.7 shows its instruction directives.

The supporting 57110/67110 microprogram controller is quite rare, its documentation even more scarce, hence the following has been extracted from [4][5][6] as a remedy.

Fig.8 depicts the internal architecture of the 57110/67110. It could directly address up to 512 words of microprogram control memory. One of a number of flag signals could be selected from the 5701/6701 4-bit slice to provide two way conditional branching at every step in a microprogram based on the value of the flag or its stored value. Four way conditional branches were also possible by using an on-chip control counter. Its generality allowed it to be used with other microprogrammable devices, while its complexity allowed it to be used standalone to simply act as an intelligent controller, microprogrammed to simply monitor system status and generate control signals.

Its basic addressing scheme was based on a microprogram counter; however, the least significant bit was handled such that a conditional branch to either location of the next even-odd address pair could be executed without the use of the microinstruction next-address field, thus often freeing that field for other uses. When this addressing scheme was employed, care had to be taken in writing and modifying microcode to ensure that all even-odd branch pairs were in fact stored in even-odd address pairs. Probably more than any other factor, the choice of whether or not to use the 67110 depended upon whether or not the designer wished to use this pairwise branching scheme.

The 67110 also included an internal looping capability not found in most equivalent devices, which allowed it to execute a program loop a specified number of times without the usual sequence of decrementing the counter, checking for zero, and branching or incrementing on the test result. It had a single-level subroutine capability, which was rather limiting. Internal storage for various processor-element array flags was provided, as was the ability to condition the next-address selection based on the stored or current value of any of those flags.

The 67110 was able to execute eight unique instructions including conditional and unconditional branches, conditional and unconditional singe-level subroutine calls, and subroutine returns, see Fig.9. A 10-bit microinstruction field included: 3 bits to control the microprogram sequence, 2 bits to control shifting, and 5 bits to select flag options where the four LS bits selected from the flag-status register bits C, N, V, Z (typically for carry, negative, overflow and zero) or control memory address register, and the fifth bit modified the selection. Although this flag field was relatively large,

this mainly reflected the flexibility in selecting the flag bits to be used in carrying out conditional operations.

Thus for the 6700 family, a microinstruction comprises an encoded l0-bit sequencer directive and an encoded 8-bit processing element directive, and various other direct and encoded fields used to control the ancillary hardware components in the system. These devices were expected to be connected in more elaborate configurations than the simple 4-bit example system shown in Fig.10, with its example microinstruction fields as in Fig.11, but this example does illustrate most of the salient points.

The 5701/6701 is described in detail in its datasheet [7]. It was also described in [8], while the 67110 was described in detail in [9] and [10], but unfortunately none of those latter documents are currently available.

Many thanks to Brian Coghlan for donating these items.

The homepage for this catalog is at: https://www.scss.tcd.ie/SCSSTreasuresCatalog/
Click *'Accession Index'* (1st column listed) for related folder, or *'About'* for further guidance.
Some of the items below may be more properly part of other categories of this catalog, but are listed here for convenience.

Accession Index	Object with Identification
TCD-SCSS-T.20250922.003	Monolithic Memories 6700 bit-slice chipset. A bipolar microcoded bit-slice processor. 1974.
TCD-SCSS-T.20250922.003.01	1 x Monolithic Memories 6701D 4-bit bit-slice arithmetic logic unit. [**at ALC**]
TCD-SCSS-T.20250922.003.02	2 x Monolithic Memories 67110D microprogram controller. [**on wirewrap prototype board**]
TCD-SCSS-X.20250916.001	Dr.Brian Coghlan's Collection of Early Microprocessors. An extensive and nearly complete set of unused 1970s microprocessor chips, most accompanied with documentation, some with demonstration boards. 1971.

References:

1. Wikipedia, *Monolithic Memories*, see:
https://en.wikipedia.org/wiki/Monolithic_Memories
Last browsed to on 22-Sep-2025.

2. Grokipedia, *Monolithic Memories*, see:
https://grokipedia.com/page/monolithic_memories
Last browsed to on 22-Sep-2025.

3. Maurice Wilkes, *The Best Way to Design an Automatic Calculating Machine*, In: *Manchester University Computer, Inaugural Conference*, July 1951, pp.16-18; reprinted in *Annals of the History of Computing*, pp.118-121, Vol.8, 1986, and in *The Early British Computer Conferences*, M.R. Williams and M. Campbell-Kelly, eds., Charles Babbage Institute Reprint Series for the History of Computing, pp.182-184, Vol.14, MIT Press, Cambridge, Mass., and Tomash Publishers, Los Angeles, 1989.

4. Phillip M. Adams, *Microprogrammable microprocessor survey*, ACM SIGMICRO Newsletter, pp.7-38, Vol.9, No.2, 1st June, 1978, see:
https://doi.org/10.1145/1096529.1096530
Also: https://www.scss.tcd.ie/SCSSTreasuresCatalog/hardware/TCD-SCSS-T.20250922.003/MicroprogrammableMicroprocessorSurvey-PhillipAdams-ACM-1Jun1978.pdf
Last browsed to on 22-Sep-2025.

5. W. Thomas Adams and Scott M. Smith, *How bit-slice families compare: Part 1, evaluating processor elements*, pp.91-98, Vol.51, No.16, Electronics, 3rd August, 1978, see:
https://www.scss.tcd.ie/SCSSTreasuresCatalog/hardware/TCD-SCSS-T.20250922.003/Electronics-1978-08-03-p91to98.pdf

Last browsed to on 22-Sep-2025.

6. W. Thomas Adams and Scott M. Smith, *How bit-slice families compare: Part 2, sizing up the microcontrollers*, pp.96-102, Vol.51, No.17, Electronics, 17th August, 1978, see:
https://www.scss.tcd.ie/SCSSTreasuresCatalog/hardware/TCD-SCSS-T.20250922.003/Electronics-1978-08-17-p96to102.pdf
Last browsed to on 22-Sep-2025.

7. Monolithic Memories, *5701/6701 4-bit Expandable Bipolar Microcontroller*, August 1974, see:
https://www.scss.tcd.ie/SCSSTreasuresCatalog/hardware/TCD-SCSS-T.20250922.003/MMI-5701-6701-MCU-datasheet-Aug1974.pdf
Last browsed to on 22-Sep-2025.

8. Clive Ghest, *5701/6701 4-Bit Expandable Bipolar Microcontroller*, Monolithic Memories Incorporated, Sunnyvale, California, 1976.

9. Clive Ghest, *A Powerful Microprogram Controller: The 67110*, Monolithic Memories Incorporated, Sunnyvale, California, 1976.

10. Monolithic Memories Incorporated, *57110/67110 Microprogram Controller*, Monolithic Memories Incorporated, Sunnyvale, California, 1977.

11. Wikimedia, *MMI 6701D die*, see:
https://commons.wikimedia.org/wiki/File:MMI_6701D_die.JPG
Last browsed to on 22-Sep-2025.

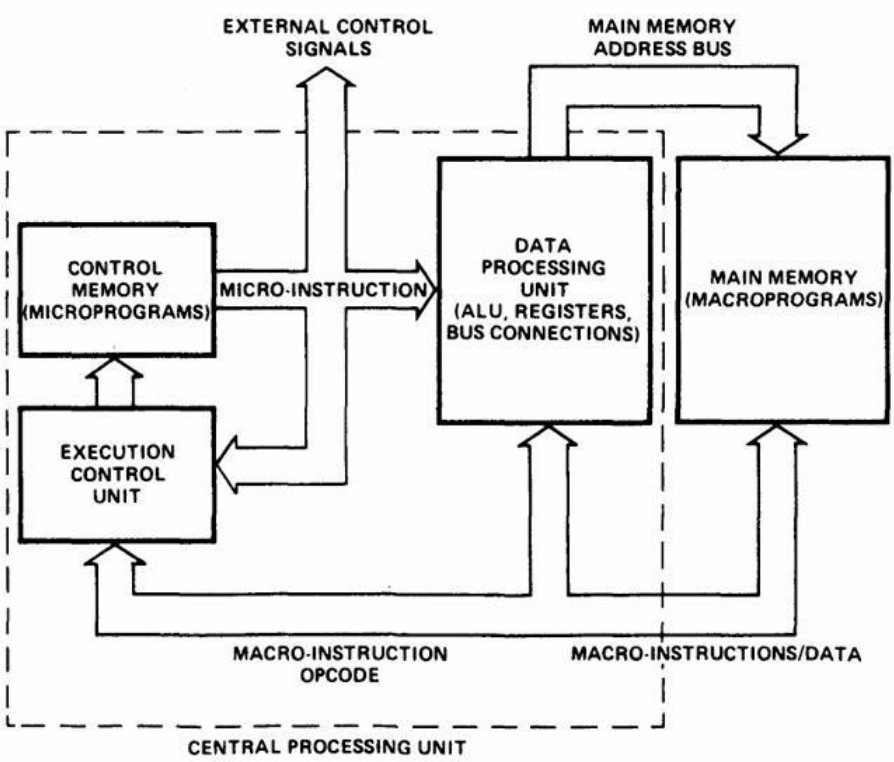


Figure 1: Microprogrammed central processing unit and user-level memory.

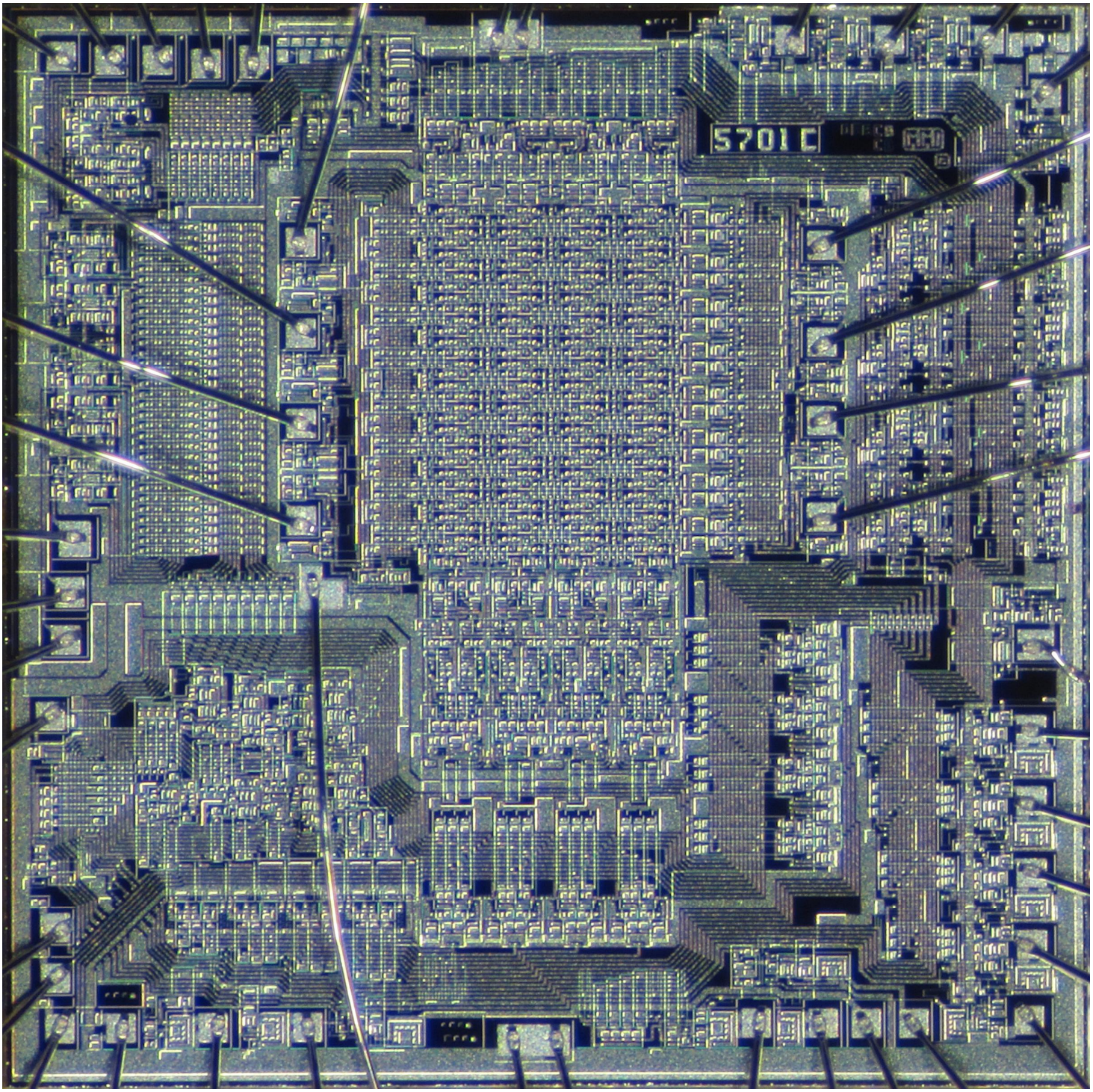
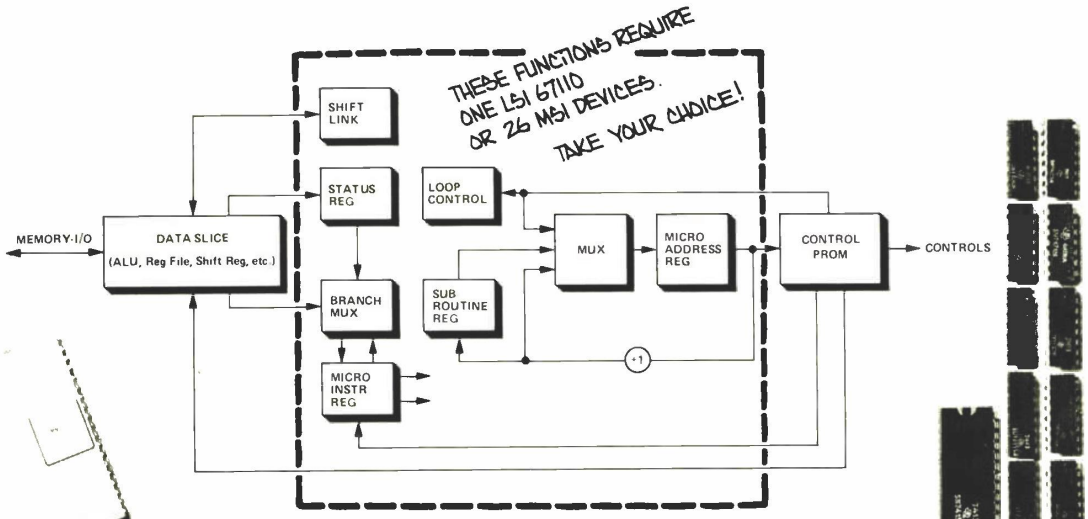


Figure 2: MMI 5701 chip die micrograph (from Wikipedia [11]).

The Only Bipolar Microprogram Controller



The 67110 Handles every control application from discs to CPUs.

And replaces 26 or more MSI devices.

Applications

CPU
Process Control
Disc Control
High Speed Printer Control
CRT Controller
Signal Processing Control

DEVICE	TEMPERATURE	100 QUANTITY PRICE
67110J	Commerical	$25.00
57110D	Military	$55.00

Features

- Works with any bit slice microprocessor such as MMI 6701, 2901, 3002.
- Works as a stand alone non-arithmetic controller
- Directly addresses 512 words of microprogram storage
- On-chip five bit loop counter for program looping routines
- Data shift linkage for arithmetic and logic shifting with 4 bit slices
- Microsubroutine and four way branch capabilities
- Very High Speed — 33 MHz

Information

For more information about this revolutionary microprogram controller and about other members of our growing family of LSI logic devices that will eventually replace all MSI logic, call, TWX or write:
In the United States,
Ed Barnett or John Birkner.
In Europe, Bernd Kruse

United States
Monolithic Memories, Inc.
1165 East Arques Avenue
Sunnyvale, CA 94086
Tel: (408) 739-3535
TWX: 910-339-9229

Europe
Monolithic Memories, GmbH
8000 Munich 80
Mauerkircherstr. 4
West Germany
Tel: (089) 982601, 02, 03, 04
Telex: (841) 524385

Far East
MMI Japan KK
Parkside-Flat Bldg
4-2-2, Sendagaya Shibuya-Ku
Tokyo 151, Japan
Tel: (3) 403-9061
Telex: (781) 26364

Monolithic Memories

Figure 3: MMI 67110 advertisement (from p.2, Electronics, 14th October, 1976).

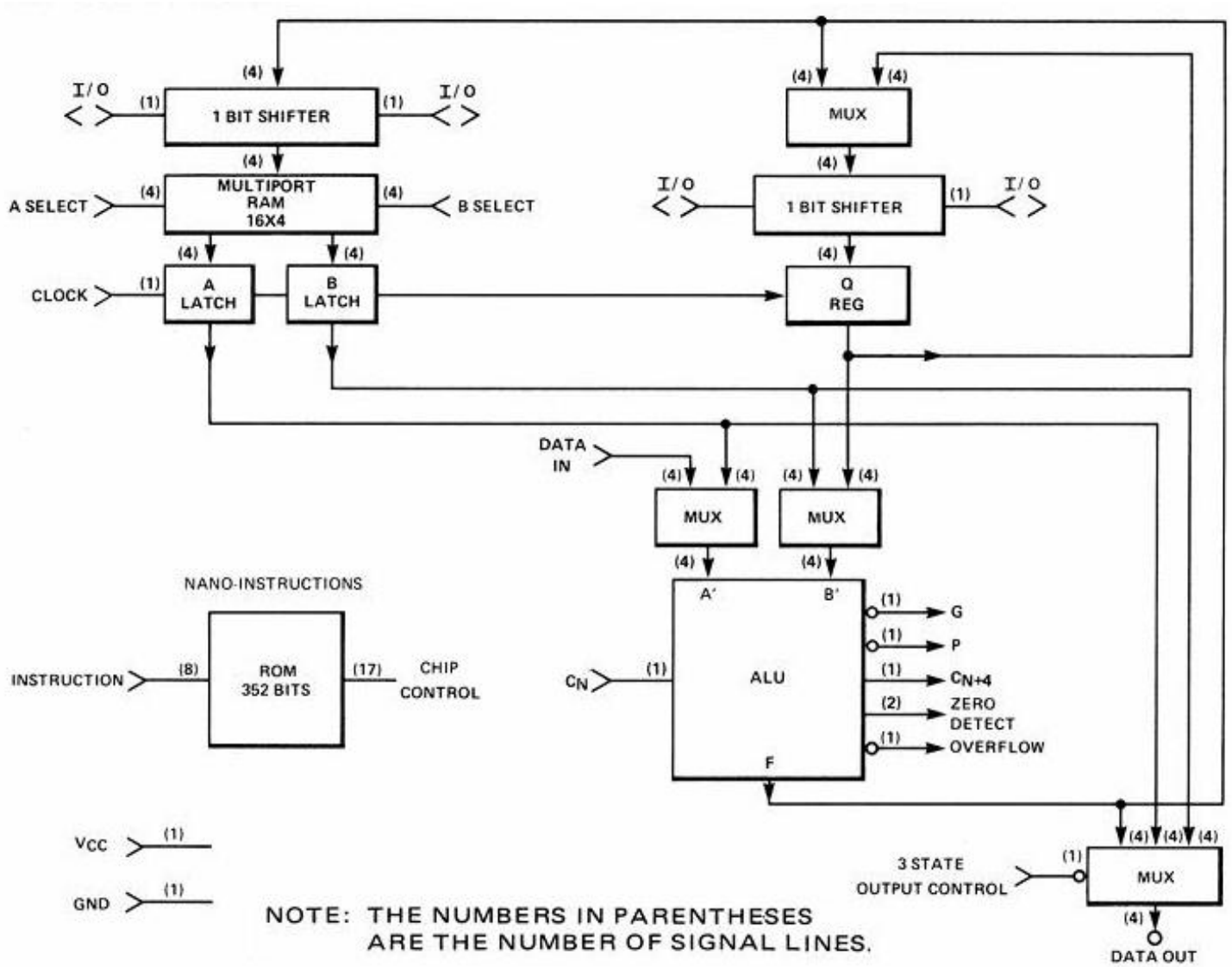


Figure 4: MMI 6701 architecture.

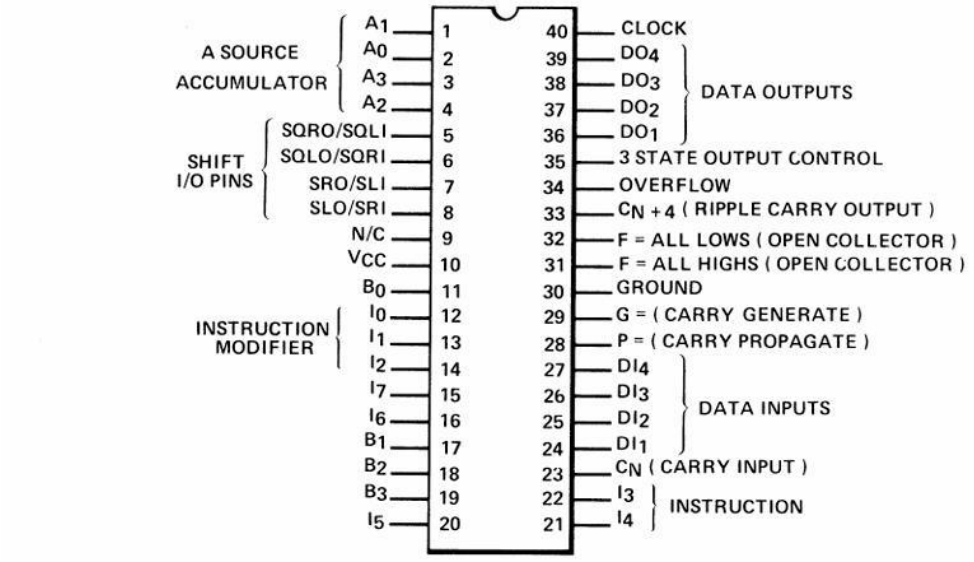


Figure 5: MMI 6701 pinout.

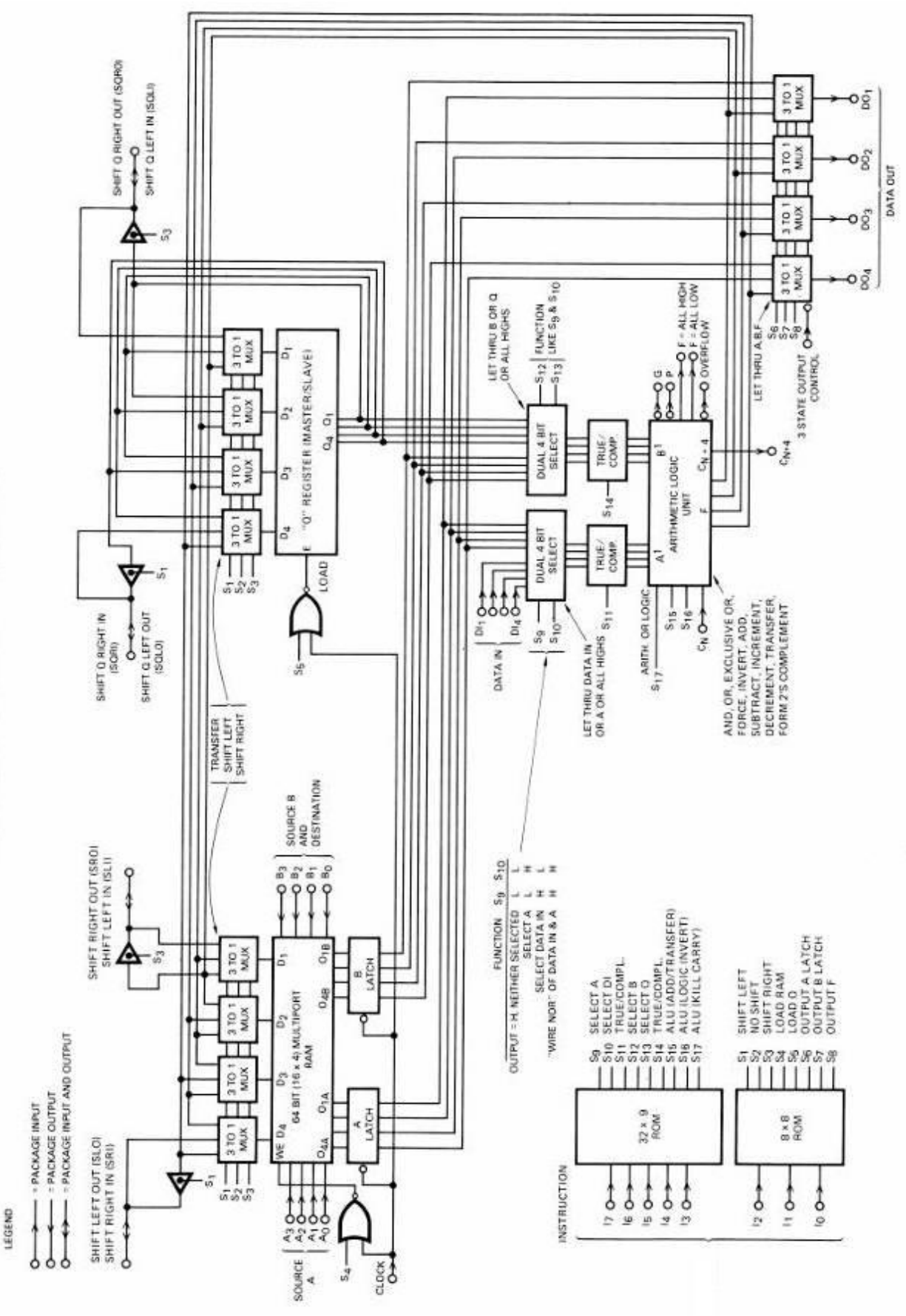


Figure 6: MMI 6701 detailed block diagram.

INSTRUCTIONS IN THE 32 x 9 ROM — POSITIVE LOGIC (1 = H ≈ 3 V) INTERPRETATION

ROM WORD							ALU Instruction (See Pg. 8 for Symbology)	ALU OUTPUT		TYPICAL USES	
I_7	I_6	I_5	I_4	I_3	Decimal	Octal		No Carry In (C_N = L)	With Carry In (C_N = H)		
L	L	L	L	L	0	00	LLLL + HHHH + C_N	Force 1111	Force 0000	Initialization (Force 1's or 0's)	
L	L	L	L	H	1	01	AND A_I & B_j	$A_I \wedge B_j$	$A_I \wedge B_j$	AND	A_I & B_j
L	L	L	H	L	2	02	AND D_I & B_j	$D_I \wedge B_j$	$D_I \wedge B_j$	"	D_I & B_j
L	L	L	H	H	3	03	OR A_I & B_j	$A_I \vee B_j$	$A_I \vee B_j$	OR	A_I & B_j
L	L	H	L	L	4	04	OR D_I & B_j	$D_I \vee B_j$	$D_I \vee B_j$	"	D_I & B_j
L	L	H	L	H	5	05	Exclusive OR A_I & B_j	$A_I \veebar B_j$	$A_I \veebar B_j$	Exclusive Or	A_I & B_j
L	L	H	H	L	6	06	Exclusive OR D_I & B_j	$D_I \veebar B_j$	$D_I \veebar B_j$	"	D_I & B_j
L	L	H	H	H	7	07	$\overline{A_I}$ + HHHH + C_N	$\overline{A_I}$ + 1111	$\overline{A_I}$	Invert	A_I
L	H	L	L	L	8	10	$\overline{D_I}$ + HHHH + C_N	$\overline{D_I}$ + 1111	$\overline{D_I}$	"	D_I
L	H	L	L	H	9	11	$\overline{B_j}$ + HHHH + C_N	$\overline{B_j}$ + 1111	$\overline{B_j}$	"	B_j
L	H	L	H	L	10	12	$\overline{Q}$ + HHHH + C_N	$\overline{Q}$ + 1111	$\overline{Q}$	"	Q
L	H	L	H	H	11	13	$\overline{A_I}$ + LLLL + C_N	$\overline{A_I}$	$\overline{A_I}$ + 0001	2's Complement Of	A_I
L	H	H	L	L	12	14	$\overline{D_I}$ + LLLL + C_N	$\overline{D_I}$	$\overline{D_I}$ + 0001	"	D_I
L	H	H	L	H	13	15	$\overline{B_j}$ + LLLL + C_N	$\overline{B_j}$	$\overline{B_j}$ + 0001	"	B_j
L	H	H	H	L	14	16	$\overline{Q}$ + LLLL + C_N	$\overline{Q}$	$\overline{Q}$ + 0001	"	Q
L	H	H	H	H	15	17	A_I + LLLL + C_N	A_I	A_I + 0001	Transfer Or Increment	A_I
H	L	L	L	L	16	20	D_I + LLLL + C_N	D_I	D_I + 0001	"	D_I
H	L	L	L	H	17	21	B_j + LLLL + C_N	B_j	B_j + 0001	"	B_j
H	L	L	H	L	18	22	Q + LLLL + C_N	Q	Q + 0001	"	Q
H	L	L	H	H	19	23	A_I + HHHH + C_N	A_I + 1111	A_I	Decrement Or Transfer	A_I
H	L	H	L	L	20	24	D_I + HHHH + C_N	D_I + 1111	D_I	"	D_I
H	L	H	L	H	21	25	B_j + HHHH + C_N	B_j + 1111	B_j	"	B_j
H	L	H	H	L	22	26	Q + HHHH + C_N	Q + 1111	Q	"	Q
H	L	H	H	H	23	27	A_I + B_j + C_N	A_I + B_j	A_I + B_j + 0001	Add	A_I & B_j
H	H	L	L	L	24	30	D_I + B_j + C_N	D_I + B_j	D_I + B_j + 0001	"	D_I & B_j
H	H	L	L	H	25	31	A_I + Q + C_N	A_I + Q	A_I + Q + 0001	"	A_I & Q
H	H	L	H	L	26	32	D_I + Q + C_N	D_I + Q	D_I + Q + 0001	"	D_I & Q
H	H	L	H	H	27	33	A_I + $\overline{B_j}$ + C_N	A_I - B_j - 0001	A_I - B_j	Subtract	A_I & B_j
H	H	H	L	L	28	34	B_j + $\overline{A_I}$ + C_N	B_j - A_I - 0001	B_j - A_I	"	B_j & A_I
H	H	H	L	H	29	35	D_I + $\overline{B_j}$ + C_N	D_I - B_j - 0001	D_I - B_j	"	D_I & B_j
H	H	H	H	L	30	36	B_j + $\overline{D_I}$ + C_N	B_j - D_I - 0001	B_j - D_I	"	B_j & D_I
H	H	H	H	H	31	37	D_I + $\overline{Q}$ + C_N	D_I - Q - 0001	D_I - Q	"	D_I & Q

INSTRUCTION MODIFIERS IN THE 8 x 8 ROM — POSITIVE LOGIC (1 = H ≈ 3 V) INTERPRETATION

Rom Word			Rom Word	Load Control		Shift Control			Data Out Control		
I_2	I_1	I_0	Decimal	Load Ram B_j	Load Q	Shift Left	Shift Right	Don't Shift	A Latch	B Latch	ALU Output F
L	L	L	0	X				X			X
L	L	H	1	X				X	X		
L	H	L	2	X				X		X	
L	H	H	3	X		X					X
H	L	L	4	X			X				X
H	L	H	5	X	X	X					X
H	H	L	6	X	X		X				X
H	H	H	7		X			X			X

Figure 7: MMI 6701 instructions.

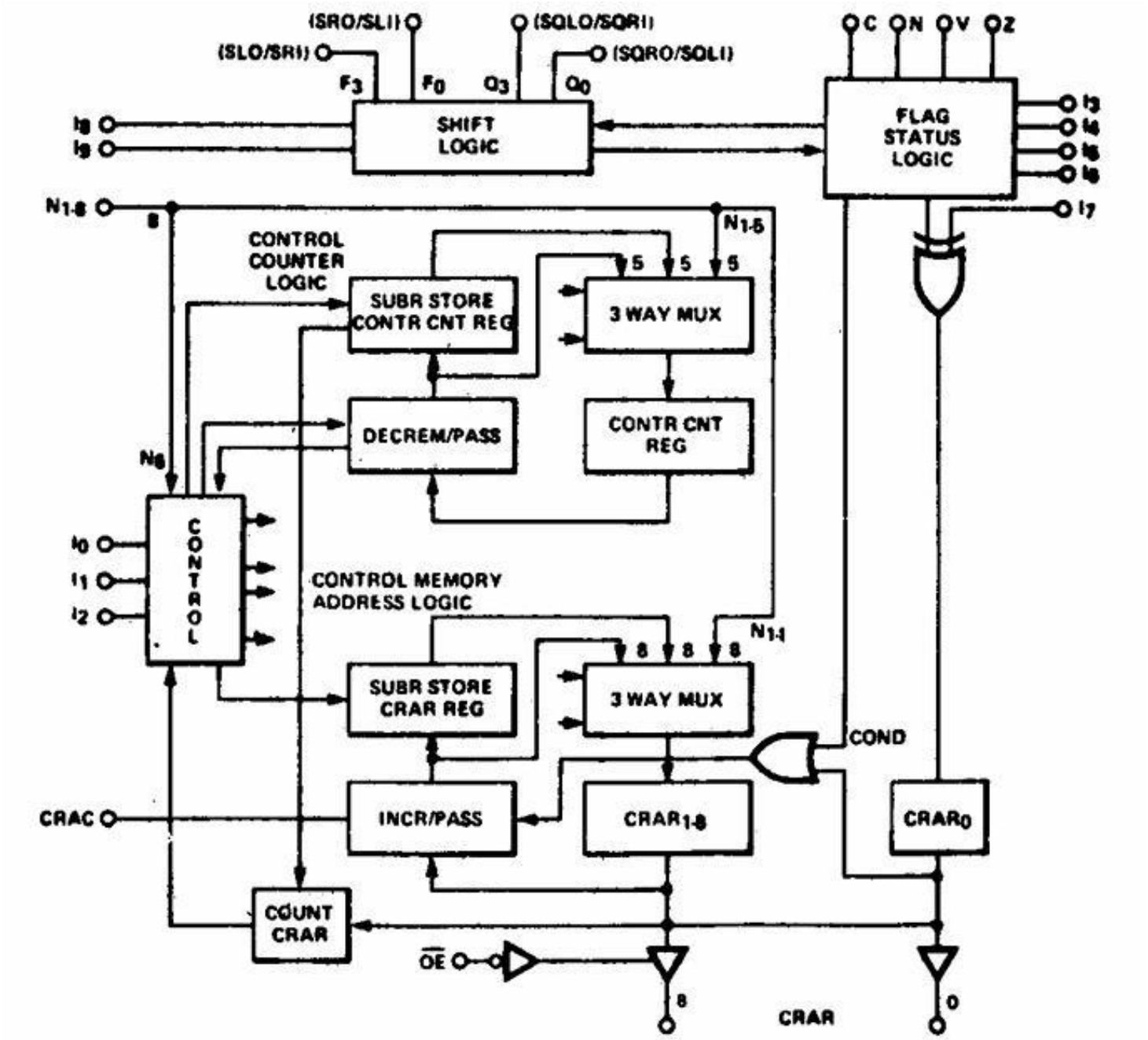


Figure 8: MMI 67110 architecture.
CRAR is the control memory address register.

Control Code			Control Action	Address Field Destination
I_2	I_1	I_0		
0	0	0	Continue to next μinstruction	None
0	0	1	Continue to next μinstruction	Control Counter, Clear SRFF if $N_6 = 1$
0	1	0	Jump to next μinstruction if Control Counter ≠ 0, Decrement Control Counter	None/CRAR (Cond. Jump)
0	1	1	Subroutine Jump to next μinstruction if Control Counter ≠ 0, Decrement Control Counter	None/CRAR (Cond. Subr. Jump)
1	0	0*	Return from Subroutine	None
1	0	1**	Jump to next μinstruction Return from Subroutine when Control Counter Subroutine Latch ≡ $CRAR_{0-4}$	CRAR (Jump Subroutine)
1	1	0	Jump to next μinstruction	CRAR (Jump)
1	1	1	Subroutine Jump to next μinstruction	CRAR (Jump Subroutine)

* The MPC 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 MPC 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.

Control Code				Action		
I_6	I_5	I_4	I_3	Operation	$CRAR_0$	Branch
0	0	0	0	None	I_7	Unconditional*
0	0	0	1	Store C	I_7	Unconditional*
0	0	1	0	Store N, V, Z	I_7	Unconditional*
0	0	1	1	Store C, N, V, Z	I_7	Unconditional*
0	1	0	0	Shift Flag Register into Q_0	I_7	Unconditional*
0	1	0	1	Shift Flag Register out of Q_0	I_7	Unconditional*
0	1	1	0	Instantaneous value of Q_0 to $CRAR_0$	$Q_0 \oplus I_7$	Conditional**
0	1	1	1	Instantaneous value of Q_3 to $CRAR_0$	$Q_3 \oplus I_7$	Conditional**
1	0	0	0	Stored value of C to $CRAR_0$	$SC \oplus I_7$	Conditional**
1	0	0	1	Stored value of N to $CRAR_0$	$SN \oplus I_7$	Conditional**
1	0	1	0	Stored value of V to $CRAR_0$	$SV \oplus I_7$	Conditional**
1	0	1	1	Stored value of Z to $CRAR_0$	$SZ \oplus I_7$	Conditional**
1	1	0	0	Instantaneous value of C to $CRAR_0$	$C \oplus I_7$	Conditional**
1	1	0	1	Instantaneous value of N to $CRAR_0$	$N \oplus I_7$	Conditional**
1	1	1	0	Instantaneous value of V to $CRAR_0$	$V \oplus I_7$	Conditional**
1	1	1	1	Instantaneous value of Z to $CRAR_0$	$Z \oplus I_7$	Conditional**

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.

SC, SN, SV, SZ are Contents of Carry, Sign, Overflow, and Zero Flip Flops.

* Incrementation of $CRAR_{1-8}$ occurs if $CRAR_0 = 1$.

** Incrementation of $CRAR_{1-8}$ always occurs.

Control Code			Bidirectional Shift Lines Acting as Outputs			
I_9	I_8	Shifting Operation	F_3 (SLO/SRI)	F_0 (SRO/SLI)	Q_3 (SQLO/SQRI)	Q_0 (SQRO/SQLI)
0	0	Arithmetic Shift Left	–	Q_3	–	Flag (SC)
0	1	Arithmetic Shift Right	$N \oplus V$	–	F_0	–
1	0	Rotate Shift Left	–	F_3	–	Flag (SC)
1	1	Rotate Shift Right	F_0	–	F_0	–

– High Impedance State

SC Contents of Carry Flip Flop

Figure 9: MMI 67110 instructions.

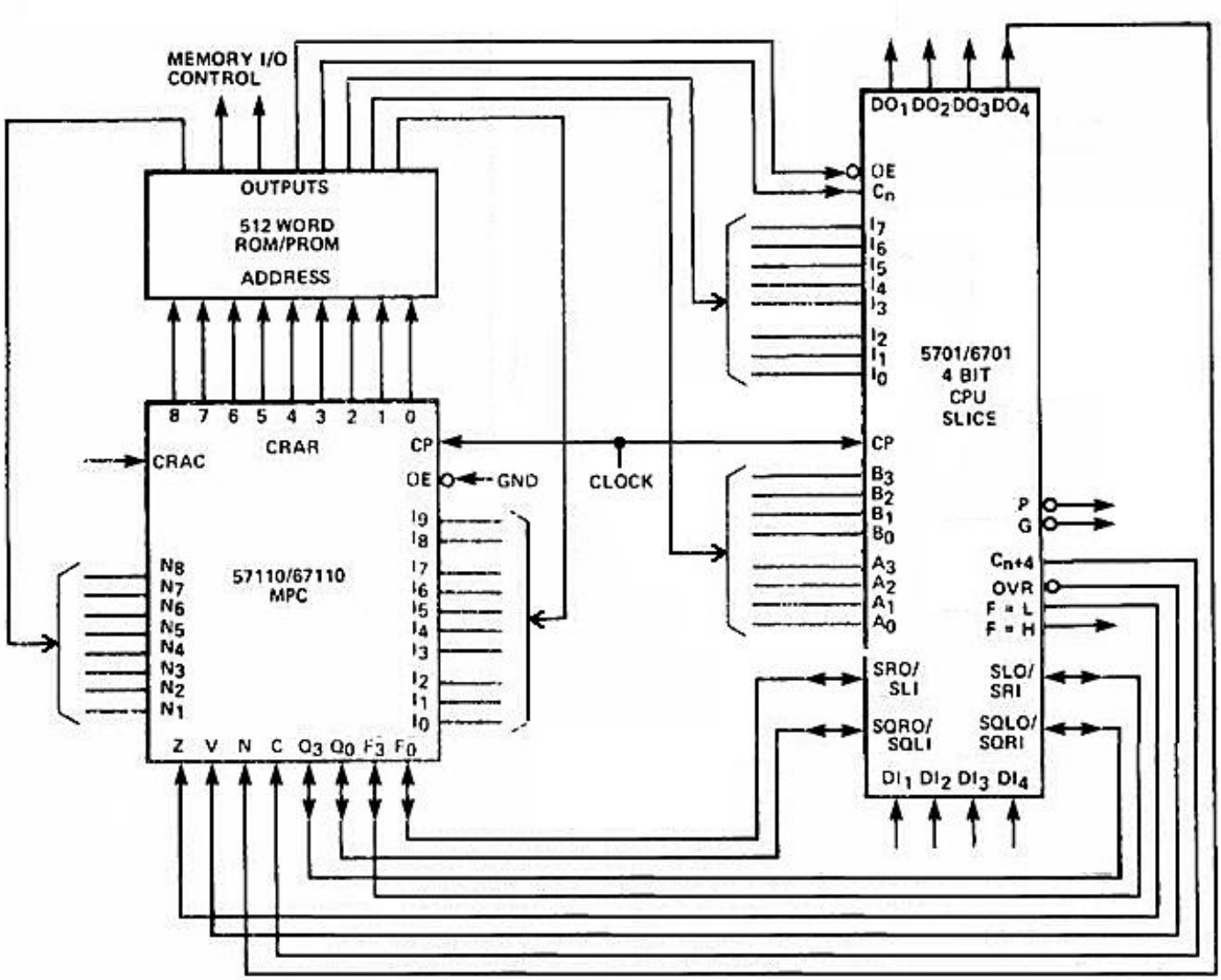


Figure 10: MMI 67110 and 6701 4-bit system configuration.

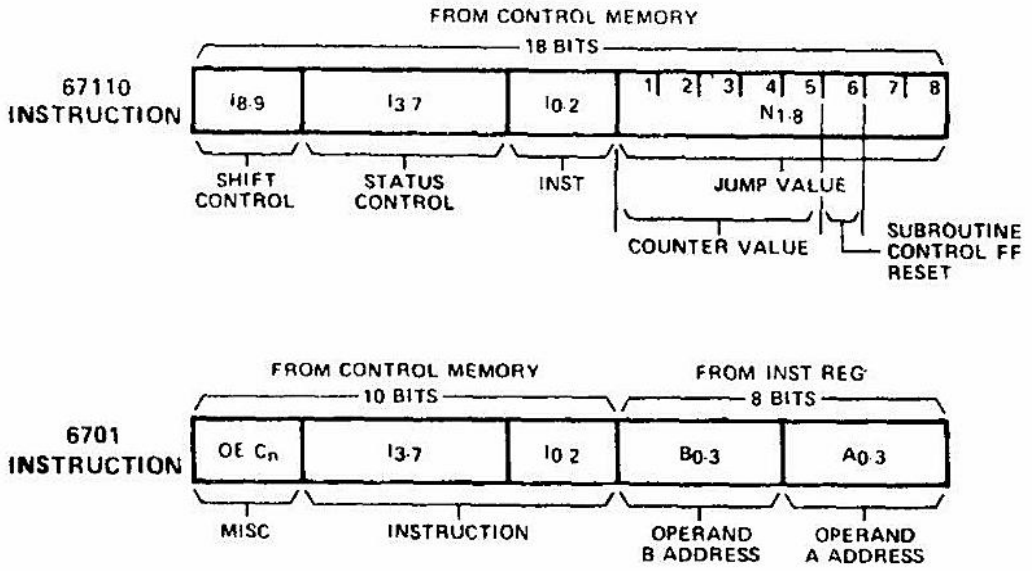


Figure 11: MMI 67110 and 6701 microinstruction fields for the system of Fig.10.