Software as Instrument: COMDASUAR and the Co-Design of Code and Hardware

Rodrigo F. Cádiz
rcadiz@uc.cl
Music Institute and Department of Electrical Engineering, Pontificia Universidad Católica de Chile
Santiago, Chile

Federico Schumacher
federico.schumacher@uchile.cl
Sound Department, Universidad de Chile
Santiago, Chile

Juan Parra Cancino
juan.parra@orpheusinstituut.be
Orpheus Instituut
Ghent, Belgium

Michel Rozas
mvrozas@uc.cl
Department of Electrical Engineering, Pontificia Universidad Católica de Chile
Santiago, Chile

Tomás Koljatic
takoljat@uc.cl
Music Institute, Pontificia Universidad Católica de Chile
Santiago, Chile

Miguel Farías
miguel.farias@uc.cl
Music Institute, Pontificia Universidad Católica de Chile
Santiago, Chile

Figure 1: Frontal view of the Comdasuar. The instrument integrates a frontal control panel populated with numerous rotary knobs, switches, and patch points for real-time manipulation of synthesis parameters, alongside a keyboard interface for symbolic input. A black and white TV was used as a visual display for the device.

Abstract

This paper contributes a detailed reverse-engineered account of the COMDASUAR (Asuar's Digital–Analog Musical Computer), a hybrid musical computer developed in Chile between the late 1970s and 1980s. Built around an Intel 8080 microprocessor and integrated with analog synthesis modules and a frontal control panel for real-time parameter manipulation, the COMDASUAR combined hardware circuitry with an extensive software environment written in assembly language. Drawing on archival access to the original instrument and to José Vicente Asuar's handwritten notebooks, we document the system's modular software organization and analyze a previously unpublished retrogradation subroutine that reverses a musical voice in memory under strict resource constraints. Through this case study we extend ongoing NIME and computer-music discussions of software as a constitutive element of digital musical instruments [17–19, 21], showing how compositional operations such as retrogradation are encoded directly as executable instrument functions, and how a corrective pass over musically meaningful control symbols (e.g., glissando and voice-end markers) makes the transformation both computational and musically informed. The COMDASUAR thus provides a historically grounded, Latin-American example of how instrumentality emerges from the co-design of code and hardware, complementing existing accounts that have largely

NIME '26, June 23–26, 2026, London, UK

centered on institutional computer-music environments in the United States and Europe.

Keywords

COMDASUAR, instrument design, software architecture, embedded musical systems, hybrid digital–analog instruments, computer-music history, Latin America

1 Introduction

The design of new musical electronic instruments routinely involves the co-development of hardware and software. Within the NIME community and adjacent computer-music scholarship, an extensive body of work has long argued that software is not a neutral implementation layer but a site where instrumental identity, idiomatic affordances, and musical thought are formed: Magnusson's account of digital instruments as "epistemic tools" [17], his later treatment of code and notation as forms of *sonic writing* [18], McPherson and Tahıroğlu's analysis of how computer-music languages shape aesthetic decisions through idiomatic patterns [19], and earlier formulations by Wessel and Wright on intimate computer-based control [21] all converge on the view that software co-constitutes what an instrument is and what it makes easy to play, compose, or imagine. Our paper does not contest this view but contributes to it, by offering a historically grounded, non-Anglo-European case in which the entanglement of code and circuitry is unusually legible.

That case is the COMDASUAR (Computador Musical Digital-Analógico Asuar), a microprocessor-based hybrid musical system designed and built by the Chilean composer and engineer José Vicente Asuar between 1975 and the early 1980s [13]. Within the COMDASUAR, structural operations such as retrogradation, canon, interpolation, and probabilistic manipulation are not applied after the fact but embedded as executable procedures within the instrument's architecture. Composition becomes inseparable from procedural logic: musical form is articulated through code, control structures, and symbolic transformations, and software operates as a structural language for musical narration that determines how musical ideas are generated, organized, and enacted in time.

Within the Latin American context of the 1970s and 1980s, marked by technological scarcity and distance from major centers of computer music, this approach is particularly significant. Asuar did not simply adopt existing technologies; he appropriated them as compositional media, intertwining musical and computational syntax. The result is a practice in which programming becomes a compositional act, and the design of executable structures becomes a way of articulating musical thought.

Asuar's interest in the use of computation in music emerged in the late 1960s. In 1971, he presented Formas 1, a work for orchestral ensemble composed using an IBM 360 computer. For this purpose, he developed a set of instructions in Fortran IV aimed at "develop[ing] a musical composition system that would be especially suitable to be solved by a computer" [2, 51]. This was his first attempt to employ computation in music from a compositional standpoint.

In 1972, he began developing the project El Computador Virtuoso, in which a PDP-8 computer, through voltage control, operated a series of MOOG synthesis modules (902, 911, 904A, 904B, 904C, 6402, and 901) as well as an ARP 2600 synthesizer. This work was primarily focused on demonstrating the possibilities of using computation in music through the recreation of selected works from the Western classical repertoire (Bach, Debussy, Ravel) using synthesized sounds. However, Asuar himself argued that the most significant step in the use of these new technologies was not the re-instrumentation of preexisting works, but rather that "the most interesting and important thing that can be done with this system is to create music" [4, 19]. This, then, became the main objective of COMDASUAR: to participate in musical creation as an active agent.

It is precisely in this shift, from technology as instrument to technology as compositional language for creating music, that COMDASUAR becomes especially relevant. Historically, this device has been studied and praised primarily for its hardware capabilities; the present paper contributes a complementary view by documenting the software side of the system in detail. Read alongside the existing software-as-instrument discourse cited above, COMDASUAR offers a concrete, archivally grounded example of how a musical instrument can be defined not only by oscillators, filters, and I/O circuits, but also by a modular software environment, symbolic command structures, and transformation routines that shape musical behavior in performance. This idea is central to Asuar's own understanding of what a musical computer should do and be [3].

The authors of this article have recently obtained access to the interior of the COMDASUAR, and have mapped most of its hardware components to date, something not previously reported in the literature until a recent publication by three of the authors of this work [20]. One of the many interesting discoveries upon opening the COMDASUAR, was the presence, among many clearly self-made arrays and circuits, of multiple instances of a mass-produced PCB, as shown in figure 2. Preliminary examination confirmed that these PCBs were the EPS 9724-1, fabricated for and sold and distributed by the DIY electronics Magazine "Elektor". This particular PCB was the Voltage Control Filter (VCF) component of a larger synthesizer project known as the "Elektor Formant". "EPS" (Elektor Printed Circuits/Project System) was a series of kits designed for the Elektor Formant, which also included the VCO (9723-1) and ADSR (9725). The 9724-1 is described as a 24dB/octave (4-pole) low-pass filter, a key component of the Formant synthesizer's sound. The design is often associated with the classic Moog-style transistor ladder filters, common in 24dB/octave circuits of that era [1].

At this moment, we cannot confirm that any (of all) of the EPS circuits are used in the COMDASUAR as part of the audio manipulation signal path, which would be its most traditional application. Given the multiple PCBs found on the inside of the device, we are inclined to believe that Asuar used these filters to tame the audio output, in order to get a cleaner output from each of its channels. In order to conduct tests to clarify this fundamental issue, we are currently in the process of acquiring modern replicas of the filter PCB, both in 5u and Eurorack formats, by makers Harald Antes[1] and Jos van Ras[2], respectively. We have also conducted sound tests with an original Elektor formant unit, and produced some test recordings of the VCF, for future reference and comparison, in case the intended use of (some of) the circuits was effectively to process audio signals.

Apart from these hardware discoveries, we have also had access to numerous hand-written notes by Asuar himself, in several paper notebooks stored in Asuar's family house in Calera de Tango in the Metropolitan Region of Chile. This has allowed us

[1] http://haraldswerk.de
[2] http://www.josvanras.com

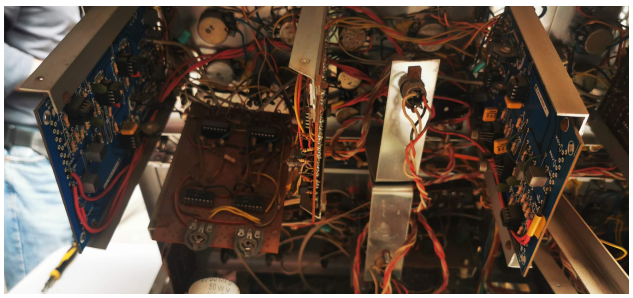

Figure 2: Interior view of the COMDASUAR revealing multiple printed circuit boards, including several instances of the Elektor EPS 9724-1 Voltage Controlled Filter (VCF). Originally designed for the Elektor Formant synthesizer system, this 24 dB/octave (4-pole) low-pass filter, associated with Moog-style transistor ladder architectures, appears integrated alongside custom-built circuitry. The coexistence of mass-produced DIY PCBs and self-fabricated modules illustrates the hybrid and resourceful construction strategy underlying the instrument's analog synthesis section.

to analyze some of Asuar's assembly code and a glimpse into his musical thinking through his computer code.

2 Hardware Overview

The COMDASUAR hardware is built upon a late-1970s Intel 8080 microcomputer architecture, integrating program and volatile memory, bus control logic, and programmable peripheral devices. Program memory is implemented using Intel 2708 EPROM devices [7], introduced in 1976, providing 1 KB (1024×8) ultraviolet-erasable storage for firmware and modular control routines. Working memory is supplied through contemporary DRAM or SRAM devices typical of the period. Bus management and signal conditioning are handled by the Intel 82212 bus controller [6], while address decoding is implemented using Intel 8205 1-of-8 decoders [8], enabling structured memory and I/O mapping across the system.

Peripheral expansion and real-time coordination are achieved through several programmable interface components. The Intel 8255 Programmable Peripheral Interface [10] provides configurable parallel ports used for digital control of the analog synthesis subsystem, including DAC interfacing and routing logic. System timing is governed by the Intel 8253 Programmable Interval Timer [11], which generates periodic interrupts and clock-derived control signals, while serial communication is implemented using the Intel 8251 USART [9]. Keyboard input is processed via the MM5740 / KR2376-57 ASCII encoder [12]. Although specific DAC models have not yet been confirmed, devices typical of the era, such as the DAC0800 [15], would convert 8-bit digital values into control voltages for voltage-controlled modules (VCO, VCF, VCA), forming a hybrid digital–analog instrument in which software-defined processes govern analog sound synthesis.

Figure 3 presents a generic block diagram derived from the hardware evidence identified to date. While this reconstruction provides a structured overview of the system architecture, significant reverse-engineering work remains to fully characterize signal paths, timing relationships, and undocumented subsystems. Current analysis indicates that audio processing itself was predominantly analog, with parameter control achieved through multiple DAC stages that generated control voltages for modules such as VCOs, VCFs, and VCAs. In parallel, digital control lines were employed to manage routing, channel selection, filtering modes, activation states, and multiplexing logic. This hybrid structure—combining digitally generated control voltages with discrete digital switching signals suggests a layered control architecture in which software defined states governed both continuous analog modulation and discrete structural reconfiguration of the synthesis chain.

3 COMDASUAR as a Software-Centered Instrument

Conceived as both a compositional tool and a performance instrument, the COMDASUAR integrated a digital control unit based on the Intel 8080 microprocessor with analog sound-generation modules. Unlike large institutional computer music systems of the same period, such as those developed at Bell Laboratories or EMS, the COMDASUAR was constructed largely from commercially available components integrated with custom-built circuitry under conditions of technological scarcity.

At its core, the COMDASUAR functioned as a programmable environment for symbolic musical manipulation and real-time sound production. Musical events were encoded as byte-level symbolic data in memory, and interpreted through modular software routines written directly in Intel 8080 machine language. The system included multiple subprograms dedicated to tasks such as voice allocation, pitch and duration processing, heuristic transformations (e.g., retrogradation, canon, interpolation), and direct control of digital-to-analog converters. These routines collectively formed a compact musical operating framework in which interpretation, transformation, and synthesis occurred within a unified execution loop.

A defining feature of the COMDASUAR was its emphasis on software-driven musical logic. Rather than relying on fixed hardware configurations for compositional procedures, Asuar implemented structural and transformational processes through assembly-level programming. This approach allowed the system to operate as a symbolic interpreter capable of dynamically modifying stored musical material, performing real-time playback, and applying algorithmic procedures under severe memory and processing constraints. In this sense, the COMDASUAR can be understood not merely as an early digital instrument, but as a self-contained computational environment for music, in which compositional thought was encoded directly into machine-level instructions [20].

A key aspect of the COMDASUAR is its *modularity*: 26 core subprograms stored in EPROM, organized into groups including command handling, data editing, heuristic procedures, real-time sound generation, and peripheral control [14]. This organization already suggests a relevant point: the instrument is a *software ecology*, not a single fixed patch.

Within that ecology, the "heuristic" layer is especially revealing for NIME concerns: it includes canonical transformations, retrogradation, probabilistic structuring, and operations that transmute pitch and rhythm [5, 14]. These are not mere utilities; they define what kinds of musical actions are easy, fast, and audible during creative work, shaping both compositional workflow and performative interaction. In the terms used by McPherson and Tahıroğlu, these subprograms constitute the *idiomatic* surface

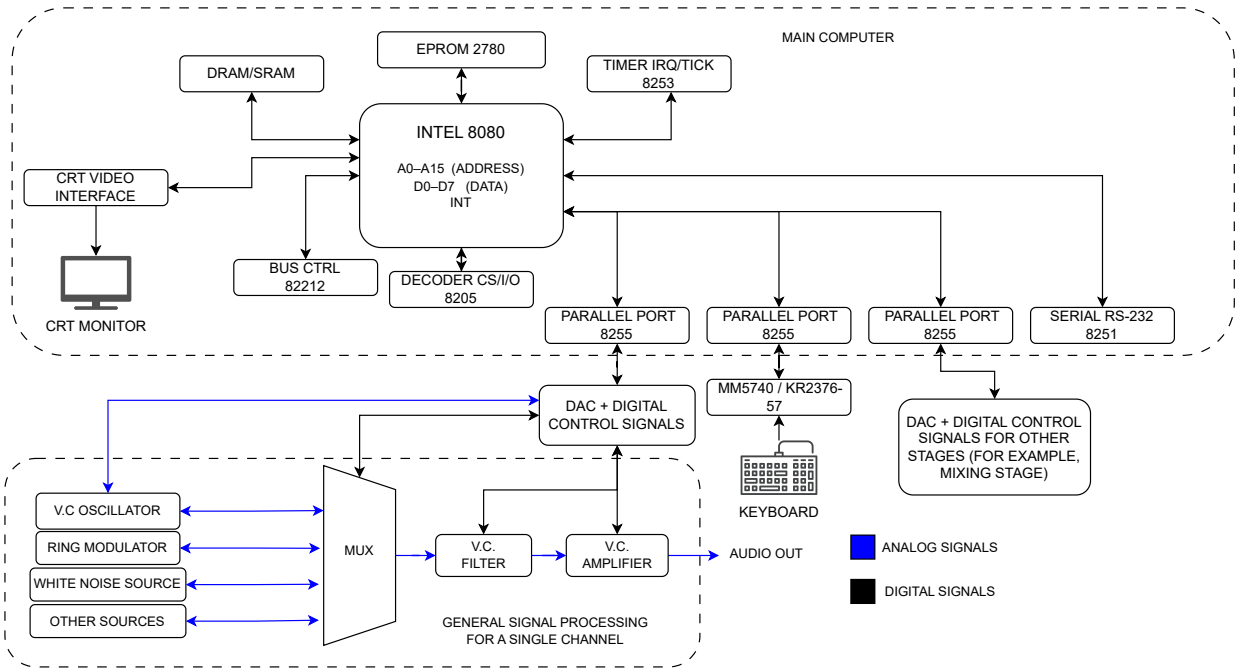

Figure 3: Generic block diagram of the Comdasuar system based on the information identified to date. This figure is an intentionally simplified, explanatory representation: it highlights the main computer (Intel 8080 subsystem) and the high-level control path used to select and drive voltage-controlled output options (e.g., VCO, ring modulator/noise sources, VCF, and VCA) through generic DAC and digital control signals. Additional subsystems are not yet included in this version, such as detailed mixing and routing stages, power-supply distribution, protection and conditioning circuitry, and other supporting analog and digital blocks.

of the instrument: the operations the system makes most readily available also tend to become the operations its users most readily think with [19].

The COMDASUAR also includes symbolic input modes designed to streamline score entry through textual codes rather than traditional notation [14]. From a NIME perspective, this is an interface design decision: it determines what counts as a gesture, how information is encoded, and how the performer/composer can navigate musical structure under real-time constraints.

3.1 How the instrument is played

A point that has remained implicit in earlier descriptions of the COMDASUAR, and that is helpful for situating the analysis that follows, concerns how the instrument was actually operated. From the documentation in [5, 14] and the notebook materials we have consulted, the working loop combines two coupled modes. In the *symbolic editing* mode, the performer/composer enters or modifies voices through textual codes typed at the keyboard, where each voice is a stream of bytes encoding pitch, duration, and a small set of musical control symbols (see Table 2), and uses command-style invocations to call subprograms from the EPROM library, including voice-data editors and the heuristic transformations (retrogradation, canon, interpolation, probabilistic operations). In the *performance/playback* mode, the frontal control panel becomes the principal expressive interface: knobs, switches, and patch points modulate synthesis parameters in real time as voices are read out and converted to control voltages by the DAC stages. Heuristic transformations such as retrogradation are therefore primarily *compositional* operations, applied to a voice in memory before or between playback passes, rather than gestures triggered note-by-note from the panel; but because the same memory is read by the playback loop, a transformation applied during a session immediately changes what the next pass will sound like. This coupling of live panel control over the analog signal path, plus on-the-fly invocation of compositional subprograms over the symbolic data is what makes it useful to describe the COMDASUAR's interface as distributed across hardware controls and software architecture rather than located in either alone.

4 Software Architecture and Instrumentality

The COMDASUAR is significant not only as a historical artifact within the development of computer-based musical systems, but also as a conceptual case that illuminates enduring questions in the field of new musical interface design and research. Its documented structure makes visible the interdependence of hardware and software in ways that resonate with longstanding NIME arguments about the constitutive role of software in digital instruments [17, 18]. Rather than presenting circuitry and control surfaces as primary and software as secondary, the COMDASUAR demonstrates how instrumentality emerges from their co-configuration. The system therefore provides a particularly lucid, archivally grounded example of how musical behavior is defined as much by program architecture as by physical components.

From this perspective, at least three fundamental questions arise; questions that continue to structure contemporary discourse in the design and study of new musical interfaces, and to which we will return explicitly in the conclusion:

(1) **Where is the interface located?** In the case of COMDASUAR, the interface is not reducible to a single layer. The instrument includes a frontal control panel equipped with numerous knobs and controls for real-time manipulation of synthesis parameters, as well as a keyboard and analog modules as shown in figure 1. These elements clearly constitute a tangible and performative interface. However, interaction is not exhausted by these visible or tactile components. It is equally mediated through symbolic codes, operational modes, and procedural transformations embedded in software (cf. Section 3.1). These symbolic structures determine how musical material is stored, interpreted, transformed, and rendered, shaping what actions are possible and how they unfold over time. The interface, therefore, must be understood as distributed across both hardware controls and software architecture, each contributing decisively to the instrument's operational and expressive identity.
(2) **How is temporal behavior constituted?** The temporal character of the instrument does not arise solely from oscillators, clocks, or analog signal paths. It is produced through control flow, synchronization routines, iteration structures, and conditional branching within software. The organization of loops, counters, and dispatch mechanisms shapes latency, rhythmic articulation, and responsiveness. Temporal experience, in this sense, is a computational as well as a physical phenomenon, a point already developed in the NIME literature on intimate computer-based control [21].
(3) **How is agency distributed?** In COMDASUAR, agency is not exclusively vested in the performer. Software routines interpret symbolic data, apply heuristic transformations, and alter operational modes during execution. Musical outcomes are therefore the result of negotiated interactions between performer input and programmed logic. This distributed configuration anticipates later discussions of autonomy, adaptive systems, and co-creative computational agents within the field.

These issues remain central to current research in new musical interfaces, particularly in contexts involving embedded systems, hybrid digital–analog architectures, and generative processes. The COMDASUAR offers an early and explicit instance in which such questions were resolved at the level of software architecture, in a setting (1970s–80s Chile) far removed from the institutional computer-music centers that have dominated existing accounts. By making these decisions visible in code structure and program organization, the system underscores a broader methodological insight: detailed documentation of instrument software, alongside circuit-level study, is necessary if NIME scholarship on historical and contemporary digital instruments is to account for software as a primary site of musical thought and interaction.

5 Example: A routine for retrogradation

Having argued that the COMDASUAR's instrumentality emerges as much from its software architecture as from its hybrid digital–analog circuitry, we now turn to a concrete example: the retrogradation subroutine for a musical voice. This routine, written by hand by Asuar in one of his recently found notebooks, is not presented here as a purely technical curiosity, nor as an isolated compositional tool, but as evidence of how musical operations are embedded directly into the instrument's software structure. Retrogradation, traditionally understood as a compositional transformation applied to symbolic material, becomes in the COMDASUAR an executable instrument function, one that operates in real time, within the same memory space as performance data. By examining how this transformation is implemented at the byte level, we can observe how compositional logic, temporal structure, and resource constraints converge in software. The routine thus illustrates the broader claim of this paper: that in COMDASUAR, and consistent with established NIME perspectives on software as instrument [17, 19], software is not merely a control layer for hardware, but a primary site where musical behavior and instrument identity are defined.

Because Asuar's voice data is itself a stream of bytes that mixes pitch/duration values with a small number of reserved *musical control symbols*, understanding the retrogradation routine requires also knowing what those reserved bytes mean. Table 2 summarizes the control bytes that appear explicitly in this routine and in the surrounding heuristic layer; the values are drawn from Asuar's notebook annotations and from the published description of the system in [5, 14]. `FE` marks the end of a voice (its terminator); `F3` marks a glissando control point; `FF` is used as an additional boundary/sentinel value. Two consequences for the routine become immediately visible: first, the forward scan that determines voice length is implemented as "walk forward until the byte under `HL` equals `FE`", and the value `FE` is therefore loaded into the accumulator before the loop precisely because it is the terminator marker; second, the corrective pass after the in-place reversal is necessary because some control symbols (notably `F3` for glissando) describe *directional* musical behavior that does not survive a literal byte-reversal unchanged, and must be re-aligned with respect to the now-reversed voice.

In the COMDASUAR notebook listings, Asuar writes programs directly in hexadecimal machine code rather than mnemonic assembly. For the Intel 8080, each hexadecimal opcode corresponds deterministically to a single instruction defined by the processor's instruction set (for example, `23h` always means `INX H`, and `7Eh` always means `MOV A,M`), as it can be consulted in the Intel 8080 assembly manual [16]. Multi-byte instructions are also fixed-length and include their operands in subsequent bytes: `CD` is always `CALL addr16` and must be followed by a 16-bit address stored little-endian, while `C2`/`CA`/`C3` are conditional/unconditional jumps followed by a 16-bit target address. This allows a reliable mechanical translation from the notebook's hex sequences to standard 8080 mnemonics: the retrogradation routine, for instance, advances pointers through the voice stream using `INX H` (`23`), detects terminators and special markers using `CMP M` (`BE`) after loading comparison constants with `MVI A,d8` (`3E`), and performs in-place reversal through byte-level exchanges using indirect memory moves (`MOV A,M` / `MOV M,A`) together with BC-indirect transfers (`STAX B` / `LDAX B`). Table 1 displays the principal mnemonics needed to understand the assembly code.

The routine shown in table 3 implements the retrogradation of a musical voice stored in memory by reversing the order of its events in place. The program begins by calling "UBICA VOZ (SET VOICE)" and "COMIENZO VOZ (START VOICE)", which together position the processor at the start of the selected voice's data stream. After skipping a small header (two "INX H" instructions),

Hex	Bytes	Mnemonic (Intel 8080)	Generic Description
CD	3	`CALL addr16`	Calls a subroutine at the specified 16-bit address. The return address is pushed onto the stack before jumping.
23	1	`INX H`	Increments the HL register pair by 1 (HL ← HL+1). Used for pointer advancement.
2B	1	`DCX H`	Decrements the HL register pair by 1 (HL ← HL-1). Used for pointer backward movement.
13	1	`INX D`	Increments the DE register pair by 1 (DE ← DE+1). Often used as a counter.
1B	1	`DCX D`	Decrements the DE register pair by 1 (DE ← DE-1).
11	3	`LXI D, d16`	Loads a 16-bit immediate value into the DE register pair.
3E	2	`MVI A, d8`	Loads an 8-bit immediate value into the accumulator (A).
7E	1	`MOV A, M`	Copies the byte from memory at address HL into the accumulator (A ← [HL]).
77	1	`MOV M, A`	Stores the accumulator into memory at address HL ([HL] ← A).
02	1	`STAX B`	Stores the accumulator into memory at the address contained in register pair BC ([BC] ← A).
0A	1	`LDAX B`	Loads the accumulator from memory at the address contained in register pair BC (A ← [BC]).
BE	1	`CMP M`	Compares the accumulator with the byte in memory at address HL. Sets flags but does not modify registers.
C2	3	`JNZ addr16`	Jumps to the specified address if the Zero flag is not set.
CA	3	`JZ addr16`	Jumps to the specified address if the Zero flag is set.
C3	3	`JMP addr16`	Unconditional jump to the specified 16-bit address.
E5	1	`PUSH H`	Pushes the HL register pair onto the stack.
E1	1	`POP H`	Pops the top of the stack into the HL register pair.
D5	1	`PUSH D`	Pushes the DE register pair onto the stack.
D1	1	`POP D`	Pops the top of the stack into the DE register pair.
C1	1	`POP B`	Pops the top of the stack into the BC register pair.

Table 1: Intel 8080 opcode-to-mnemonic mapping

it preserves the starting address on the stack and initializes a counter register pair ("DE") to zero. The accumulator is loaded with the value "FE", which functions as a terminator marker for

Byte (hex)	Musical control symbol
`F3`	Glissando marker. Indicates a directed pitch transition between adjacent events; requires re-interpretation after retrogradation (see `CORRIGE GLISS`).
`FE`	End-of-voice (terminator). Used by forward-scan loops to determine the effective length of a voice in memory.
`FF`	Boundary / sentinel value used in voice and parameter tables.

Table 2: Reserved musical control bytes referenced by the retrogradation routine, as documented in Asuar's notebooks and [5, 14].

Address	Hex Code	Label / Comment
1000	CD F6 06	UBICA VOZ
	CD 3E 07	COMIENZO VOZ
	23 23	
	E5 E5	
	11 00 00	
	3E FE	
1C0F	13 23 BE	
1C12	C2 0F 1C	
	1B	
	3E FF	
	CD 6B 07	COMPARA
	23 E5 C1	
	E1	
1C1F	D5 E5	
1C21	7E 02	
1C23	23 7E	
	2B 77 23	
	CD 87 1C	CONTADOR
	C2 23 1C	
	0A 77	
1C30	E1 D1	
	CD 87 1C	CONTADOR
	C2 1F 1C	
	E1 E5	
1C3A	23	
	3E F3 BE	
	CA 62 1C	CORRIGE GLISS
1C41	3E FE BE	
	CA 4A 1C	CORRIGE F0
	C3 3A 1C	

Table 3: A Comdasuar's voice retrogradation subroutine written by hand by Asuar in one of his notebooks

the voice (see Table 2). The program then enters a forward scan loop, incrementing both the pointer ("HL") and the counter ("DE") until it encounters the "FE" byte. In this way, it determines the effective length of the voice in memory and locates its endpoint.

Once the end of the voice has been identified, the program prepares the pointers necessary for reversal. The routine "COMPARA (COMPARE)" is called, likely to compute or validate the mirrored position corresponding to the start of the voice. Through a sequence of stack manipulations ("PUSH"/"POP"), the program establishes two active pointers: one pointing to the beginning of the voice and another to its end. With these in place, the routine enters its core retrogradation loop. Using "MOV", "STAX", and "LDAX", it exchanges bytes between the forward and backward positions. A counter managed by the subroutine "CONTADOR

(COUNTER)" determines when half of the sequence has been swapped, ensuring that the process stops once the central point of the voice has been reached.

The swap logic operates directly at the byte level, indicating that the voice is encoded as a stream of symbolic musical events. Rather than constructing a new reversed copy, the program performs the retrogradation in place, overwriting memory positions as it progresses. This approach minimizes memory usage, which is an important consideration in the constrained RAM environment of the Intel 8080, and reflects an economical programming style. Each iteration of the loop updates pointers and counters, gradually exchanging mirrored elements until the voice has been completely reversed. The use of conditional jumps ("JNZ") ensures that execution remains tightly synchronized with the swap counter.

After the main reversal phase, the program performs a corrective pass through the now-retrograded data. It scans for the special marker bytes `F3` and `FE` (Table 2), which correspond to musical control symbols: a glissando marker and a voice-end indicator, respectively. When such values are detected, execution branches to routines labeled "CORRIGE GLISS (CORRECT GLISSANDI)" or "CORRIGE F0 (CORRECT F0)", which adjust these control codes to preserve musical coherence after reversal: a glissando whose direction is implicit in the order of surrounding pitches must be re-tagged once that order is inverted; a terminator that originally sat at the end of a voice must be reasserted in its new position. This final stage demonstrates that the retrogradation is not purely mechanical but musically aware: certain symbolic elements require reinterpretation when temporal direction is inverted. The program thus combines low-level memory manipulation with higher-level musical logic, embodying Asuar's characteristic approach of encoding compositional thinking directly into assembly language.

6 Architectural Implications for Instrument Design

The retrogradation routine discussed above provides more than a technical illustration of assembly programming; it offers a revealing case of how musical operations become architectural features of an instrument. Rather than existing as an abstract compositional concept, retrogradation is realized as an executable procedure embedded in the operational logic of the system. This embedding has broader implications for how instrument design is understood, particularly in hybrid digital–analog environments.

First, the routine demonstrates that software architecture can function as instrument architecture. The retrogradation procedure does not operate as an external utility but as an integrated subroutine within a modular program structure that includes voice selection, boundary detection, counter management, and corrective passes. Its existence presupposes a broader software environment organized into callable units, dispatch mechanisms, and state management strategies. In this sense, modular program organization is not merely a matter of code efficiency; it defines what transformations are readily available within performance and compositional workflows. The architecture of subprograms becomes part of the instrument's structural identity, an observation consistent with the more general claim that programming languages and environments shape the idiomatic surface of computer-music instruments [19].

Second, the implementation of retrogradation foregrounds temporal behavior as a design decision. The routine operates within the same memory space and execution loop that governs real-time playback. Its logic, first forward scanning to detect a terminator, in-place reversal using dual pointers, and then termination based on a counter reaching the midpoint, reflects a precise handling of time and iteration under constrained resources. Temporal structure is therefore not solely the product of oscillators or clocks, but of control flow, branching conditions, and loop boundaries. The character of musical transformation emerges directly from computational sequencing.

Third, the corrective pass applied after reversal reveals that symbolic layers within the system are musically meaningful rather than neutral data. The detection of special marker bytes (e.g., glissando or boundary indicators) and their subsequent adjustment indicates that the system distinguishes between structural data and expressive control symbols. Retrogradation is thus not a purely mechanical inversion but a musically informed transformation. This illustrates how symbolic encodings and transformation procedures co-constitute the instrument's expressive logic.

Taken together, these aspects suggest that in COMDASUAR, compositional operations are not external manipulations applied to a finished instrument. They are embedded within the system's executable structure. The retrogradation routine exemplifies how software procedures, memory organization, and control logic define the range and character of possible musical actions. Instrumentality, in this context, arises from the co-design of hardware circuitry and computational architecture, with software serving as a primary site where musical thought is formalized and enacted.

7 Conclusion

This study has contributed a partial reverse-engineered account of the COMDASUAR, a hybrid digital–analog system built around an Intel 8080 microprocessor and an extensive software environment in Assembly, drawing on archival access to the original instrument and to Asuar's handwritten notebooks. Read alongside existing NIME and computer-music scholarship that already treats software as a constitutive element of digital instruments [17–19, 21], the COMDASUAR provides a historically specific, Latin-American case in which the entanglement of code and circuitry is unusually well documented and unusually visible. The system's identity is shaped not only by its oscillators, control panel, and analog modules, but equally by its modular subprogram structure, symbolic input mechanisms, and real-time execution logic; the analysis of the retrogradation routine, in particular, makes this co-configuration concrete by showing a traditionally compositional transformation realized as an executable instrument function operating directly on memory, complete with a musically aware corrective pass over reserved control bytes (Tables 3, 2).

It is worth returning explicitly to the three questions raised in Section 4, and stating what the COMDASUAR case has allowed us to say about each.

Where is the interface located? On the evidence of both the hardware survey and the performance loop described in Section 3.1, the interface of the COMDASUAR is genuinely distributed: the frontal control panel governs the analog signal path in real time, while symbolic codes entered at the keyboard, together with subprograms invoked from the EPROM library, govern what is in memory to be played at all. Neither layer is reducible to the other, and a description of the instrument that

omits the software side—as much existing writing on the COMDASUAR has done—loses a substantial portion of what users actually interact with.

How is temporal behavior constituted? The retrogradation routine makes the answer concrete. The temporal structure of a transformed voice is not produced by oscillators or clocks alone; it is produced by a forward scan that locates a terminator byte, by dual pointers walking inward, by a counter detecting the midpoint, and by conditional jumps that decide when to stop. Latency, articulation, and the very direction of musical time are properties of control flow as much as of analog signal flow.

How is agency distributed? In the COMDASUAR, the performer/composer chooses which voice to operate on, when to invoke a heuristic transformation, and how to modulate the analog output through the panel; but the transformations themselves carry musically informed decisions of their own, which are most clearly in the corrective passes that re-interpret glissando and boundary markers after a reversal. Outcomes therefore arise from a negotiation between performer input and programmed logic, and the line between "what the performer did" and "what the instrument did" is drawn inside the software.

Situating COMDASUAR within Asuar's earlier compositional context further clarifies why this distributed configuration is not incidental. A work such as *Buffalo 71*, composed precisely in 1971 and circulated in international experimental music environments, illustrates an earlier phase in which technological processes already operated not only as means of sound production but as structural components of compositional thought. COMDASUAR radicalizes that direction by integrating programming into the compositional process itself: code becomes part of the structural narrative of the composer, and musical thinking is articulated through executable procedures, symbolic encoding, and control logic. Recognizing software as instrumental material does not diminish the importance of hardware design; rather, and consistently with prior NIME arguments [17, 19], it clarifies that instrumentality arises from the co-configuration of circuitry and code. In this sense, COMDASUAR stands as a particularly transparent, archivally documented example, and one whose Latin-American provenance complements the more familiar institutional histories of computer music, of how code and circuitry together constitute the musical instrument.

Ethics Statement

This research consisted of the technical and historical analysis of a musical instrument and its associated documentation. No human participants and no animal subjects were involved at any stage of the study; consequently, no procedures requiring informed consent or institutional ethical approval were necessary.

The authors declare no conflicts of interest related to this work, and the study was conducted independently and without commercial sponsorship influencing the design, analysis, or interpretation of results. Access to the COMDASUAR instrument and to the handwritten notebooks of José Vicente Asuar was granted by the persons currently responsible for the instrument and the related archival materials, in accordance with applicable access permissions and customary archival practices. The authors are not descendants of Asuar and have no proprietary claim over his work; all photographic and documentary materials reproduced here are used with permission of the relevant custodians and for the scholarly purposes of this research, with reasonable efforts made to ensure accurate representation of the instrument and its components. No additional ethical concerns are known to arise from this work.

References

[1] Formant: The elektor music synthesiser. *Elektor*, 32:27–34, Dec. 1977. December 1977.

[2] J. V. Asuar. Música con computadores ¿cómo hacerlo? *Revista Musical Chilena*, 26(118):36–66, 1972.

[3] J. V. Asuar. Haciendo música con un computador. *Revista Musical Chilena*, 27(123-124):81–82, 1973.

[4] J. V. Asuar. Recuerdos. *Revista Musical Chilena*, 29(132):5–22, 1975.

[5] J. V. Asuar. Un sistema para hacer música con un microcomputador. *Revista Musical Chilena*, 34(151):5–28, 1980.

[6] I. Corporation. *Intel 82212 8-Bit Bus Controller Datasheet.* Intel Corporation, 1975.

[7] I. Corporation. *Intel 2708 UV Erasable Programmable Read-Only Memory Datasheet.* Intel Corporation, 1976.

[8] I. Corporation. *Intel 8205 One-of-Eight Decoder Datasheet.* Intel Corporation, 1976.

[9] I. Corporation. *Intel 8251 USART Datasheet.* Intel Corporation, 1977.

[10] I. Corporation. *Intel 8255A Programmable Peripheral Interface Datasheet.* Intel Corporation, 1977.

[11] I. Corporation. *8253 Programmable Interval Timer Datasheet.* Intel Corporation, 1978.

[12] S. M. Corporation. *MM5740 / KR2376 ASCII Keyboard Encoder Datasheet.* SMC, 1979.

[13] C. Fuentes and F. Schumacher. José Vicente Asuar y el COMDASUAR: Un caso de composición musical cognitivamente extendida. *Epistemus. Revista de estudios en Música, Cognición y Cultura*, 9(2):42–68, 2021.

[14] M. A. Fumarola. Report of the comdasuar: A significant and unknown chilean contribution in the history of computer music. In *Proceedings of the International Computer Music Conference (ICMC)*, Ann Arbor, Michigan, 1998.

[15] T. Instruments. *DAC0800 8-Bit Digital-to-Analog Converter Datasheet.* Texas Instruments, 1999.

[16] Intel Corporation. *Intel 8080 Microcomputer Systems User's Manual.* Intel Corporation, Santa Clara, CA, 1975.

[17] T. Magnusson. Of epistemic tools: Musical instruments as cognitive extensions. *Organised Sound*, 14(2):168–176, 2009.

[18] T. Magnusson. *Sonic Writing: Technologies of Material, Symbolic, and Signal Inscriptions.* Bloomsbury Academic, New York, 2019.

[19] A. McPherson and K. Tahıroğlu. Idiomatic patterns and aesthetic influence in computer music languages. *Organised Sound*, 25(1):53–63, 2020.

[20] F. Schumacher, J. Parra, and R. F. Cadiz. The mind amplifier: Software composition in josé vicente asuar's comdasuar. *Organised Sound*, 31(1):11–20, 2026.

[21] D. Wessel and M. Wright. Problems and prospects for intimate musical control of computers. *Computer Music Journal*, 26(3):11–22, 2002.

Acknowledgments

This research was funded by Avanza UC grant AV25334 from Vicerrectoría de Investigación, Pontificia Universidad Católica de Chile, and by ANID Núcleo Milenio Animupa NCS2025_12. Anthropic's Claude 3.5 Sonnet was used to assist with computer code analysis, language editing, grammar improvement, and writing style refinement. The authors reviewed and edited the final content and take full responsibility for the article's content. The authors would like to thank Claudio and Malise Asuar, descendants of José Vicente Asuar, for granting us access to both the COMDASUAR and Jose Vicente's handwritten notes.