**Group Name: Binary Cartel**

**Project Title: Health Monitoring System**

**Week 6: Instruction Set Architecture and Documentation**

**Course: CS3520**

**Date: 19/10/25**

# Overview and Motivation

A straightforward RISC-style instruction set designed for tiny health monitoring devices is the **BCHMS-32 (Binary Cartel Health Monitoring System ISA)**. It facilitates the collection, sorting, filtering, and storing of sensor data by the device. Moving averages, real-time heart rate checks, and simple health condition tests all use the ISA's support for both integer and floating-point operations.

## Design Goals:

• Simple hardware implementation.  
• Efficient execution of real-time signal processing.  
• Easy pipelining and predictable performance.  
• Compact encoding suitable for low-power processors.

## Suitability

The **BCHMS-32** performs math and memory operations fast and effectively, making it a good choice for health monitoring. Additionally, it keeps the hardware low-power and small, which makes it appropriate for devices that must process data on a regular basis.

# Architectural Design Choices

Instruction Philosophy: HMS-32 adopts a RISC-style approach to minimize complexity and maximize performance. RISC simplifies decoding and enables efficient pipelining—critical for continuous sensor data processing.

Registers: BCHMS-32 defines 16 general-purpose integer registers (R0–R15), 8 floating-point registers (F0–F7), and 5 special-purpose registers (Program Counter, Stack Pointer, Frame Pointer, Status Register, Instruction Register). This configuration provides enough fast storage for arithmetic, filtering, and classification tasks.

Data Types: BCHMS-32 supports 32-bit integers and 32-bit floating-point values, aligning with the sensor data and moving average calculations found in the target workload.

Addressing Modes: The ISA provides immediate, register, base+offset, PC-relative, and post-increment addressing. These modes support array processing, sensor data access, and control flow efficiently.

Memory Model: BCHMS-32 uses a flat 32-bit address space, little-endian ordering, and enforces 4-byte alignment for data and instructions to improve memory access efficiency.

Instruction Formats: BCHMS-32 uses a fixed 32-bit instruction format for all instructions. Fixed-length encoding simplifies decoding and ensures predictable timing.

# Instruction Set Summary

|  |  |  |
| --- | --- | --- |
| Category | Mnemonic(s) | Description |
| Arithmetic | AR\_ADD, AR\_SUB, AR\_MUL, AR\_DIV, AR\_ADI | Integer arithmetic and immediate operations |
| Logical | LG\_AND, LG\_OR, LG\_XOR, LG\_SHL, LG\_SHR | Logical and bit manipulation operations |
| Floating Point | FP\_ADD, FP\_SUB, FP\_MUL, FP\_DIV, FP\_ITF, FP\_FTI | Floating-point arithmetic and conversion |
| Memory | MM\_RDW, MM\_WRW, MM\_RDB, MM\_WRB, MM\_MOV | Load, store, and register movement |
| Branch | BR\_EQ, BR\_NE, BR\_GT, BR\_LT, BR\_JMP, BR\_CAL, BR\_RET | Control flow and conditional branching |
| Compare | CF\_CMP, CF\_TST | Comparison and flag setting |
| System/I-O | IO\_INP, IO\_OUT, IO\_WRF, IO\_TRP | I/O operations and system calls |

# Instruction Formats in BCHMS-32

The BCHMS-32 ISA employs four related 32-bit fixed formats that correspond to the kinds of operations present in the health-monitoring program:  
  
• R-Type (Register–Register) – used for arithmetic and logical operations between registers.  
 sum += data[i+j]; → AR\_ADD R4, R4, R1  
  
• I-Type (Immediate/Memory) – used for operations with constants or memory offsets such as array indexing and threshold comparison.  
 Example: accessing data[i] or checking if (val > threshold).  
  
• J-Type (Jump/Branch) – used for loops and conditional flow in the C++ code (for, if).  
 Example: loop back in the moving-average function.

• F-Type (Floating-Point) – specialized for the double/float calculations in the moving-average and classification sections.  
 Example: sum / windowSize; → FP\_DIV F1, F2, F3

Each format retains the 32-bit fixed length for uniform decoding and pipeline alignment. The R- and F-formats handle register arithmetic, I-format covers immediates and loads/stores, and J-format governs control flow—precisely matching the behavior of the high-level health-monitoring workload.

# Instruction Encoding

HMS-32 uses a single 32-bit fixed instruction format. The format supports both integer and floating-point operations, with opcode fields in the upper bits for fast decoding.

Field Layout (MSB → LSB):

[31–26] Opcode (6 bits) | [25–22] Rd (4 bits) | [21–18] Rs1 (4 bits) | [17–14] Rs2 (4 bits) | [13–8] Funct (6 bits) | [7–0] Immediate/Reserved (8 bits)

R-Type Example: opcode(6) | rd(4) | rs1(4) | rs2(4) | funct(6) | res(8)  
I-Type Example: opcode(6) | rd(4) | rs1(4) | imm(18)

J-Type Example: [31–26] Opcode | [25–22] Condition Reg | [21–0] PC-Relative Offset  
F-Type Example: [31–26] Opcode | [25–23] Fd | [22–20] Fs1 | [19–17] Fs2 | [16–8] Funct | [7–0] Reserved

Opcode assignments are grouped by function families: Arithmetic (0x00), Immediate (0x01), Load/Store (0x02), Branch (0x03), Jump/Call (0x04), Floating Point (0x05), and System (0x3F).

# Design Rationale and Tradeoffs

Simplicity vs Capability: HMS-32 balances simplicity with sufficient capability for health monitoring tasks. It omits complex addressing and variable-length instructions to simplify the datapath and control logic.

Code Density vs Performance: Using fixed 32-bit instructions improves performance predictability and pipelining at the cost of slightly larger code size—acceptable in this domain.

Hardware Impact: The reduced instruction set simplifies decoding and control unit design, supporting a clean 5-stage pipeline (Fetch, Decode, Execute, Memory, Writeback).

Extensibility: The opcode space and uniform instruction format allow future expansion for additional operations, such as digital signal processing or network communication extensions.

# References

1. RISC-V International, The RISC-V Instruction Set Manual, Volume I: User-Level ISA, v20191213.  
2. World Health Organisation, Global Strategy on Digital Health 2020–2025, Geneva: WHO, 2021.  
3. Abebe, R. et al., Narratives and Counternarratives on Data Sharing in Africa, arXiv:2103.01168, 2021.  
4. Okolo, C.T. et al., Responsible AI in Africa—Challenges and Opportunities, Springer, 2022.