# CS3520 ISA Specification

Team: Binary Cartel

Project Title: Health Monitoring System

Course: CS3520 – Computer Organisation and Architecture I

Week: 6 – ISA Definition & Documentation

Date: 18 October 2025

## Overview & Motivation

The Health Monitoring ISA (HM-ISA) is a customized variant of the RISC-V RV32IMC instruction set, optimized for low-power, AI-assisted mobile health monitoring. The processor targets applications such as vital-sign tracking, anomaly detection, and secure data transmission — workloads that combine frequent arithmetic operations, lightweight ML inference, and continuous data logging.  
  
Design Goals  
• Energy efficiency: Operate within mobile device constraints while performing continuous monitoring.  
• AI readiness: Include simple multiply-accumulate instructions for ML inference.  
• Compactness: Utilize compressed instructions for code density and faster fetch cycles.  
• Reliability: Maintain simple control logic to ensure predictable timing and easy verification.  
  
Suitability  
This ISA directly supports sensor-based arithmetic, vector operations, and low-bandwidth communication, making it ideal for rural and mobile-health contexts where resources are limited.

## Architectural Design Choices

|  |  |  |
| --- | --- | --- |
| Aspect | Design Decision | Rationale |
| Instruction Philosophy | Pure RISC with fixed 32-bit instructions + 16-bit compressed subset (RV32IMC) | Simplifies pipeline and control while supporting compact code storage |
| Registers | 32 general-purpose 32-bit registers (x0–x31) + 1 Status/Alert Register (SAR) | Enables efficient arithmetic and event tracking |
| Data Types | 8-bit, 16-bit, 32-bit integers; limited 32-bit fixed-point ops | Sufficient for sensor data and ML inference without FPU overhead |
| Addressing Modes | Register, Immediate, Base + Offset | Simple and efficient for load/store and branching |
| Memory Model | Little-endian, word-aligned access | Compatible with embedded systems |
| Pipeline Structure | 5 stages: IF → ID → EX → MEM → WB | Classic RISC design with good throughput and manageable hazards |
| Instruction Formats | Fixed 32-bit formats (R, I, S, B, U, J) + 16-bit compressed (C) | Keeps decode logic regular and reduces fetch energy |
| Interrupts | Basic external interrupt + software interrupt for alerts | Supports emergency notifications and sensor anomalies |

## Instruction Set Summary

|  |  |  |  |
| --- | --- | --- | --- |
| Category | Mnemonic | Operands | Description |
| Arithmetic | ADD, SUB, MUL, DIV, AVG | Rd, Rs1, Rs2 | Basic arithmetic; AVG for signal smoothing |
| Logic | AND, OR, XOR, NOT | Rd, Rs1, Rs2 | Standard bitwise operations |
| Load/Store | LD, ST | Rd/Rs, [Addr] | Load/store from memory |
| Control | BEQ, BNE, JMP, CALL, RET | Rs1, Rs2, Imm | Branch and subroutine control |
| AI Extension | VMAC, VAVG | Rd, Rs1, Rs2 | Vector multiply-accumulate and averaging |
| Security/Comms | ENC, DEC, SND, RCV | Rd, Rs1, Imm | Simple XOR-based encryption + data send/receive |
| System | HALT, INT, SETAL | — / Imm | Stop, trigger interrupt, or set alert flags |

## Instruction Encoding Summary

32-bit Formats (Standard RISC-V)  
• R-Type: | funct7 (7) | rs2 (5) | rs1 (5) | funct3 (3) | rd (5) | opcode (7) |  
• I-Type: | imm[11:0] | rs1 (5) | funct3 (3) | rd (5) | opcode (7) |  
• S-Type: | imm[11:5] | rs2 (5) | rs1 (5) | funct3 (3) | imm[4:0] | opcode (7) |  
• C-Type (Compressed): 16-bit variant with smaller immediates and register fields.  
  
Custom Extensions  
• VMAC and VAVG use R-type encoding with funct7 = 1010000x.  
• ENC and DEC use I-type encoding with opcode = 1010111.  
• Immediate fields remain 12 bits for uniformity.  
  
The fixed 32-bit width (and optional 16-bit compressed form) ensures simple decoding, low control complexity, and consistent instruction timing.

## Design Rationale & Trade-offs

|  |  |
| --- | --- |
| Consideration | Discussion |
| Simplicity vs Capability | The RV32IMC base keeps the ISA lightweight and implementable within course scope, while the custom extensions (AI + Security) provide domain-specific functionality. |
| Code Density vs Performance | The C extension reduces instruction memory by up to 25%, trading slightly slower decode for improved fetch efficiency. |
| Hardware Impact | A 5-stage pipeline with IMC extensions adds minimal control overhead. The multiplier/divider unit is straightforward to integrate into EX stage; compressed instruction decoding occurs during IF/ID. |
| Extensibility | The ISA can later include floating-point (“F”) or vector (“V”) extensions if heavier ML workloads emerge. |
| Domain Fit | Supports all target workloads: filtering (AVG), ML inference (VMAC), encryption (ENC/DEC), and health alerts (INT/SETAL). |

## 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.