

Fourth Laboratory Assignment – ZIP Code to State Code Lookup

CS130 (Section 15046)

Donaven Bruce

November 18, 2025

OVERVIEW

This assignment implements an ARMv7 (Cortex-A9) function:

- `findZips` – receives a 5-digit ZIP code in R0 and returns the corresponding FIPS state code in R0 (or 0 if invalid, out-of-range, or not found).

Execution was performed on CPULator (DE1-SoC): <https://cpulator.01xz.net/?sys=arm-de1soc>. Results were read directly from register R0.

TEST PLAN & EXPECTED RESULTS

Input ZIP	Rationale	Expected R0
90210	Valid ZIP (Beverly Hills)	6 (California)
-5	Negative number	0
0	Zero	0
333	In-range but not in any table entry	0
100000	Greater than 99999	0
91371	Pierce College (assignment example)	6 (California)
55552	Minnesota example	27 (Minnesota)

TEST RESULTS

The screenshot displays a debugger interface with the following components:

- Top Bar:** Running status and navigation buttons (Step Into, Step Over, Step Out, Continue, Stop, Restart, Reload, File, Help).
- Registers Panel:** Lists registers r0 through r18, cpsr, and spsr. Register r0 is highlighted with the value 00000006.
- Disassembly Panel:** Shows assembly code with addresses, opcodes, and disassembled instructions. The instruction at address 00000000 is highlighted: `ldr r0, [pc, #108] ; 0x74`.
- Messages Panel:** Displays compilation errors and warnings, including "Error: invalid constant -- 'mov R0, #0done:'" and "Warning: cannot find entry symbol _start; defaulting to 00000000".
- Devices Panel:** Shows the status of various hardware components:
 - LEDs: ff200000
 - Switches: ff200040
 - Push buttons: IRQ 73, ff200050
 - Seven-segment displays: ff200020
 - JTAG UART: IRQ 80, ff201000
 - Cortex-A9 Private Timer: IRQ 29, fffec000
 - Cortex-A9 Watchdog Timer: IRQ 30, fffec020
 - HPS L4 Watchdog Timer: IRQ 203, ffd02000
 - HPS L4 Watchdog Timer: IRQ 204, ffd03000
 - Interval Timer: IRQ 72, ff202000
 - Interval Timer: IRQ 74, ff202020
 - VGA pixel buffer: c0000000
 - Carworld UART: IRQ 75, ff211020

Figure 1: Valid ZIP 90210 → R0 = 6 (California)

The screenshot displays a debugger interface with several panels:

- Registers:** Shows the state of various registers. The `r0` register is highlighted with a red background and contains the value `00000000`. Other registers like `r1` through `r18` also show `00000000`. Special registers `cpsr` and `spsr` are also visible.
- Disassembly (Ctrl-D):** Shows the assembly code being executed. The address `00000000` is selected. The code includes instructions like `andeq r0, r0, r0`, `mov r0, #0`, and `bl findZips`. A comment `@ max index` is present.
- Devices:** A sidebar on the right showing various hardware components:
 - LEDs:** A row of 8 LEDs, all currently off.
 - Switches:** A row of 8 switches, all currently off.
 - Push buttons:** A row of 4 buttons, all currently off.
 - Seven-segment displays:** A row of 4 displays, all showing `000000`.
 - JTAG UART:** A section for JTAG and UART communication.
 - Cortex-A9 Private Timer:** A timer with a value of `0`.
 - Cortex-A9 Watchdog Timer:** A watchdog timer with a value of `0`.
 - HPS L4 Watchdog Timer:** Two watchdog timers, both with a value of `2147483648`.
 - Interval Timer:** Two interval timers, both with a value of `6249999`.
 - VGA pixel buffer:** A display showing a noisy, pixelated image.
 - Carworld UART:** A section for Carworld UART communication.
- Messages:** A log at the bottom showing compilation and linking messages. It includes warnings about missing entry symbols and successful compilation results.

Figure 2: Negative input $-5 \rightarrow R0 = 0$

The screenshot displays a debugger interface with the following components:

- Registers:** A list of registers (r0-r15, sp, lr, pc, cpsr, spsr) with their current values. The PC register is highlighted at address 00000008.
- Disassembly (Ctrl-D):** A table showing the disassembly of instructions at various memory addresses. The instruction at address 00000008 is `andeq r0, r0, r0`, which sets R0 to 0.
- Messages:** A log of system messages, including compilation warnings and successful completion of the assembly process.
- Devices:** A panel on the right showing the status of various hardware components:
 - LEDs:** A row of 8 LEDs, all currently off.
 - Switches:** A row of 8 switches, all currently off.
 - Push buttons:** A row of 4 push buttons, all currently off.
 - Seven-segment displays:** A row of 4 seven-segment displays, all showing 0.
 - JTAG UART:** A section for JTAG and UART communication, showing Read FIFO and Write FIFO counts.
 - Cortex-A9 Private Timer:** A section for the Cortex-A9 Private Timer, showing the current value and status.
 - Cortex-A9 Watchdog Timer:** A section for the Cortex-A9 Watchdog Timer, showing the current value and status.
 - HPS L4 Watchdog Timer:** A section for the HPS L4 Watchdog Timer, showing the current value and status.
 - Interval Timer:** A section for the Interval Timer, showing the current value and status.
 - VGA pixel buffer:** A section for the VGA pixel buffer, showing a noisy image.
 - Carworld UART:** A section for the Carworld UART, showing the current value and status.

Figure 3: Zero input \rightarrow R0 = 0

The screenshot displays a debugger interface with the following components:

- Registers:** A list of CPU registers (r0-r15, sp, lr, cpsr, spsr) with their current values. Register **r0** is highlighted in red, showing the value **00000000**.
- Disassembly (Ctrl-D):** A window showing the assembly code at the current address (00000000). The code includes instructions like `andeq r0, r0, r0`, `ldr r0, [pc, #108]`, and `bl 0xc (0xc: findZips)`. The instruction at address 00000000 is `andeq r0, r0, r0`.
- Messages:** A window showing system messages. A warning from the linker is visible: `Link: arm-eabi-ld --script build_arm.ld -e _start -u _start -o work/asmKQ6aiy.s.elf work/asmKQ6aiy.s.o`. The message states: `bin/arm-eabi-ld: warning: cannot find entry symbol _start; defaulting to 00000000`. Below this, it says "Compile succeeded." and "Code and data loaded from ELF executable into memory. Total size is 864 bytes."
- Devices:** A sidebar on the right showing various hardware components:
 - LEDs:** A row of 8 LEDs, all currently off.
 - Switches:** A row of 8 switches, all currently off.
 - Push buttons:** A row of 4 push buttons, all currently off.
 - Seven-segment displays:** A row of 4 seven-segment displays, all showing "00000000".
 - JTAG UART:** A section for JTAG UART communication, showing Read FIFO: 0 and Write FIFO: 0.
 - Cortex-A9 Private Timer:** A section for the Cortex-A9 Private Timer, showing IRQ 29 and TO=0.
 - Cortex-A9 Watchdog Timer:** A section for the Cortex-A9 Watchdog Timer, showing IRQ 30 and TO=0.
 - HPS L4 Watchdog Timer:** A section for the HPS L4 Watchdog Timer, showing IRQ 203 and INTR=0.
 - Interval Timer:** A section for the Interval Timer, showing IRQ 72 and TO=0.
 - VGA pixel buffer:** A section for the VGA pixel buffer, showing a noisy, pixelated image.
 - Carworld UART:** A section for the Carworld UART, showing IRQ 75 and a state of "Steer: 0" and "Accel: 0".

Figure 5: Out-of-range 100000 \rightarrow R0 = 0

The screenshot displays a debugger interface with the following components:

- Top Bar:** Running, Step Into (F2), Step Over (Ctrl-F2), Step Out (Shift-F2), Continue (F3), Stop (F4), Restart (Ctrl-R), Reload (Ctrl-Shift-L), File, and Help menus.
- Registers Panel:** Shows registers r0 through r18, cpsr, and spsr. Register r0 is highlighted with the value 00000006.
- Disassembly Panel:** Shows assembly code starting at address 00000000. The code includes instructions like `andeq r0, r0, r0`, `ldr r0, [pc, #108]`, and `bl findZips`. The instruction at address 00000008 is highlighted: `b 0x8 (0x8: Stop)`.
- Devices Panel:** Contains various hardware components:
 - LEDs:** ff200008
 - Switches:** ff200040
 - Push buttons:** IRQ 73 ff200058
 - Seven-segment displays:** ff200020
 - JTAG UART:** IRQ 80 ff201008
 - Cortex-A9 Private Timer:** IRQ 29 fffec680
 - Cortex-A9 Watchdog Timer:** IRQ 30 fffec620
 - HPS L4 Watchdog Timer:** IRQ 203 ffd02008
 - Interval Timer:** IRQ 72 ff202008
 - VGA pixel buffer:** c8000000
 - Carworld UART:** IRQ 75 ff211020
- Messages Panel:** Shows compilation and linking messages, including warnings about missing entry symbols and successful compilation results.

Figure 6: Pierce College ZIP 91371 → R0 = 6 (California)

The screenshot displays the OpenOCD GUI interface. The top toolbar includes buttons for 'Stopped', 'Step Into', 'Step Over', 'Step Out', 'Continue', 'Stop', 'Restart', and 'Reload'. The main window is divided into several panes:

- Registers:** Shows the state of various registers (r0-r15, sp, lr, pc, cpsr, spsr) with their current values.
- Disassembly (Ctrl-D):** The central pane showing the assembly code. It includes a search bar for 'Go to address, label, or register:' and a table of instructions with addresses, opcodes, and disassembled code. The code includes labels like `_start:`, `findZips:`, and `loop:`.
- Devices:** A sidebar on the right showing various hardware components and their status, including LEDs, switches, push buttons, seven-segment displays, JTAG UART, Cortex-A9 Private Timer, Cortex-A9 Watchdog Timer, HPS L4 Watchdog Timer, Interval Timer, VGA pixel buffer, and Carworld UART.
- Messages:** The bottom pane showing the output of the compilation and linking process. It includes warnings about missing entry symbols and successful compilation results.

Figure 7: Minnesota ZIP 55552 \rightarrow R0 = 27

CHECKLIST

- Function named exactly `findZips`
- Correct range validation (> 0 and < 100000)
- 61-entry table with proper labels and 12-byte stride
- Linear search returns state code when $\text{low} \leq \text{ZIP} \leq \text{high}$
- Returns 0 for all invalid cases
- Safe return using `BX LR`
- All required tests pass perfectly on CPULator

All requirements met – ready for submission.