# ARM Simulator Poster

CP216 – Introduction to Microprocessors

Spring 2025

Ahmed Fahim Mostafa

Jose Inigo Domingo - Joshua Ehikhuemen - Manvir Virk - Daniela Chizhevskiy

# ARMv7 Simulator

**We built an ARM simulator that works like a real processor by reading and running some basic 32-bit ARM instructions. Our goal is to understand how machine instructions manipulate registers, memory, and program control using a simplified model.**

### *Key Components*

#### Register File

16 general-purpose registers: R0–R15 (R15 is the Program Counter)

Condition flags supported: N: Negative, Z: Zero, C: Carry, V: Overflow

update_flags() sets flags based on results of ADD, SUB, CMP, and MOV

#### Instruction Decode and Execute

Supports over 15 ARM instructions, such as: MOV, ADD, SUB, CMP, LDR, STR, B

Immediate values use a special method called ROR() for rotation

Executes instructions by modifying registers, memory, and PC

### Memory Model

4096 bytes of byte-addressable memory

The memory is word-aligned, meaning all addresses are aligned to 4 bytes

read_word(address) → Reads 4 bytes starting at the given word-aligned address

write_word(address, value) → Writes 4 bytes starting at the given word-aligned address

## *How to Use the Simulator*

1.Prepare a test binary in Python:

```python
import struct

# Sample instructions: MOV R1, #4; ADD R2,
instructions = [
    0xE3A01004,  # MOV R1, #4
    0xE2812003,  # ADD R2, R1, #3
    0xE5812004,  # STR R2, [R1, #4]
    0xE5913004   # LDR R3, [R1, #4]
]

with open("test.bin", "wb") as f:
    for instr in instructions:
        f.write(struct.pack("<I", instr))
```

2.Run the simulator

python simulator.py

3.Simulator will print decoded instructions, execution steps, and the final CPU state:

```
=== Final CPU State ===
R0: 0x0
R1: 0x5
R2: 0x8
R3: 0x4e28120
R4: 0x0
```

....

```
R13: 0x0
R14: 0x0
R15: 0x10
Flags: {'N': 0, 'Z': 0, 'C': 0, 'V': 0}
```

### Design Highlights

- Used a modular design, splitting the simulator into clear components: RegisterFile, Memory, Decoder, and Executor, making the system easier to understand and manage
- The simulator shows each instruction as it runs, which makes it easier to see what's happening and fix any problems
- Status flags are updated just like in real hardware for operations such as ADD, SUB, and CMP, making the simulation more realistic
- It's also easy to extend and add new instructions if needed, making only small adjustments to the decoder and executor

### References

Introduction to Microprocessors and Architectures Slides - ARM Architecture Slides -

ARM Architecture Reference Manual - https://student.cs.uwaterloo.ca/~cs452/docs/ts7200/arm-architecture.pdf