COSMIC Functional Size of ARM Assembly Programs

Ahmed Darwish and Hassan Soubra

The German University in Cairo (GUC), Egypt
In this presentation, we:

1. **Motivate** our work, and choice of use case.
2. Provide an **overview** about COSMIC and ARM.
3. Describe the **methodology**.
4. Showcase our **prototype**.
5. **Conclude**.
Why COSMIC for Assembly languages?

- **All** languages, whether interpreted or compiled, are bound to be represented in **some assembly language** to be run by the hardware.
- We can leverage this fact to build a universal **language-agnostic** COSMIC measurement tool.
- How? Just measure the assembly language programs!

![Diagram of software compilation process](Courtesy of zentut.com)
Why ARM?

- ARM is a very big player in the semiconductor industry, licensing its chip designs to manufacturers:
  - It accounts for a third of the addressable market.
  - ARM chips are used in 90% of chips in the mobile industry.
  - 75% of vehicle infotainment and ADAS systems are ARM-based.
  - Apple recently announced incorporating ARM designs into its computers.
  - Recently acquired by NVidia!
- ARM’s architecture is RISC.
In this presentation, we:

1. **Motivate** our work, and choice of use case.
2. **Provide an overview** about COSMIC and ARM.
3. Describe the **methodology**.
4. Showcase our **prototype**.
5. **Conclude**.
ARM Architecture

- ARM processors come in 3 main flavours: Application, Real-Time, and Microcontroller.
- Three variants of the ISA exist: A32, A64, T32.
- 31 registers present in the register file, of which 16 are user addressable.
- A defining characteristic of the ISAs: a condition field which defines the state of the conditional flags that must be present for the command to run, otherwise it is discarded NOP.
COSMIC Overview

- Methodology standard for quantifying functional user requirements. (ISO 19761)
- Focuses mainly on the transfer of data groups between the different functional processes, and possibly a persistent storage.
- A functional process is initiated by a triggering event from some functional user causing a triggering entry of data into the process.
- Data movements are classified into Entries, Exits, Reads, and Writes.

D'Avanzo et al.: COSMIC functional measurement of mobile applications and code size estimation
Stages of COSMIC

COSMIC’s Measurement Manual ver. 4.0.2
In this presentation, we:

1. **Motivate** our work, and choice of use case.

2. Provide an **overview** about COSMIC and ARM.

3. Describe the **methodology**.

4. Showcase our **prototype**.

5. **Conclude**.
The Measurement Strategy Phase

- The goal of our procedure is measuring the size of compiled programs.
- We consider each instruction to be a separate functional process, since we are working at the hardware level of a computer.
- The Decoder part of the processor is our only functional user.
- The persistent storage consists of several components: the register file, caches, co-processor register files, memory attached to the processor.
The Mapping Phase

- **Triggering Event**: a fetched program is decoded and the different parameters are retrieved.
- **Triggering Entry**: the condition field, since it is common for all instructions.
- Afterwards, the different parameters and signals necessary for the execution of the instruction are passed to the functional process. All those are considered **Entries**.
- Any data exchange with the **Persistent Storage** mentioned before is considered our **Reads/Writes**.
- No **Exits**!
The Mapping Phase

- The Abstract Instruction Model

```java
void instruction(halfbyte conditionField, {boolean S}, param1, param2, ...
{
    // Optional status register check
    if (statusRegister[conditionBits] != conditionField)
        return;

    // Details of the instruction go here

    // Optional status register update
    // (for arithmetic instructions only)
    if (S == true)
        updateStatusRegister()

    return;
}
```
The ADC Example

```c
void ADC(halfbyte conditionField, boolean S, int rd, int rn, int Operand2) { ← 5 Entries
    if(statusRegister[conditionBits] != conditionField) ← 1 Read
        return; (optional)
    tmp a = RegisterFile[rt]; ← 1 Read
    RegisterFile[rd] = a + Operand2 + statusRegister[CarryFlag]; ← 1
        Write, 1 Read
    if(S == true)
        updateStatusRegister(); ← 1 Write (optional)
    return;
}
```
The PUSH Example

```c
void PUSH(halfbyte conditionField, short reglist) { ← 2 Entries
  if (statusRegister[conditionBits] != conditionField) ← 1 Read
    return;
    (optional)

  for i = 0 till 15:
    if reglist[i] == 1:
      Memory[address] = RegisterFile[i] ← 1 Read,
                              1 Write (per register)
      address = address + 4

    RegisterFile[SP] = RegisterFile[SP] - 4*BitCount(reglist); ← 1 Read,
                         1 Write

  return;
}
```
In this presentation, we:

1. **Motivate** our work, and choice of use case.
2. Provide an **overview** about COSMIC and ARM.
3. Describe the **methodology**.
4. **Showcase our prototype**.
5. **Conclude**.
Automated Measurement Tool Prototype

- A **Python** GUI tool for measuring the **size** of ARM programs.
- Compatible with the output of **objdump** command from Linux.
- Can include native C headers into the measurement.
Automated Measurement Tool Prototype
In this presentation, we:

1. **Motivate** our work, and choice of use case.
2. Provide an **overview** about COSMIC and ARM.
3. Describe the **methodology**.
4. Showcase our **prototype**.
5. **Conclude**.
Conclusion

- Our goal in this work was to map an assembly language’s computational model as comprehensively as possible to COSMIC’s terminology, in order to measure programs in that language.
- We decided to choose ARM as the target language due to its popularity and significant share in the market.
- We implemented our mappings as a simple prototype that takes as input C program/ARM assembly program.
- Possible future work: 1. Applying our mapping to other assembly languages.
  2. Studying how the size changes as a program is compiled.
Thank you!

Any Questions?

Reach out to us!

amfa.darwish.97@gmail.com

hassan.soubra@guc.edu.eg