COSMIC Functional Size of ARM Assembly Programs

Ahmed Darwish and Hassan Soubra

The German University in Cairo (GUC), Egypt

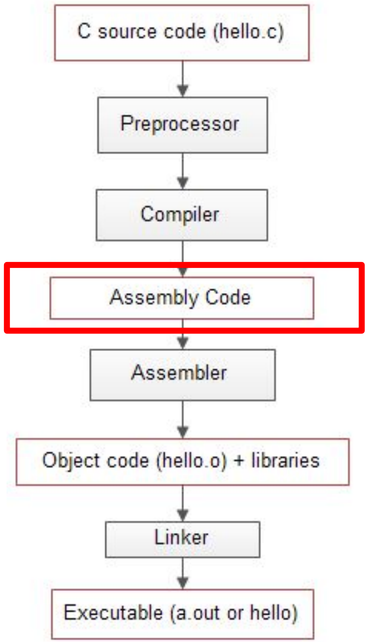
Presentation Agenda

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!



We would like to measure this!

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.

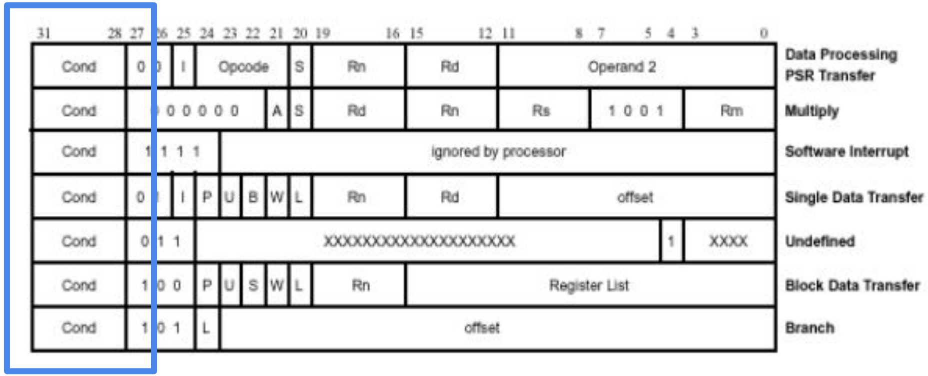
Presentation Agenda

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**.

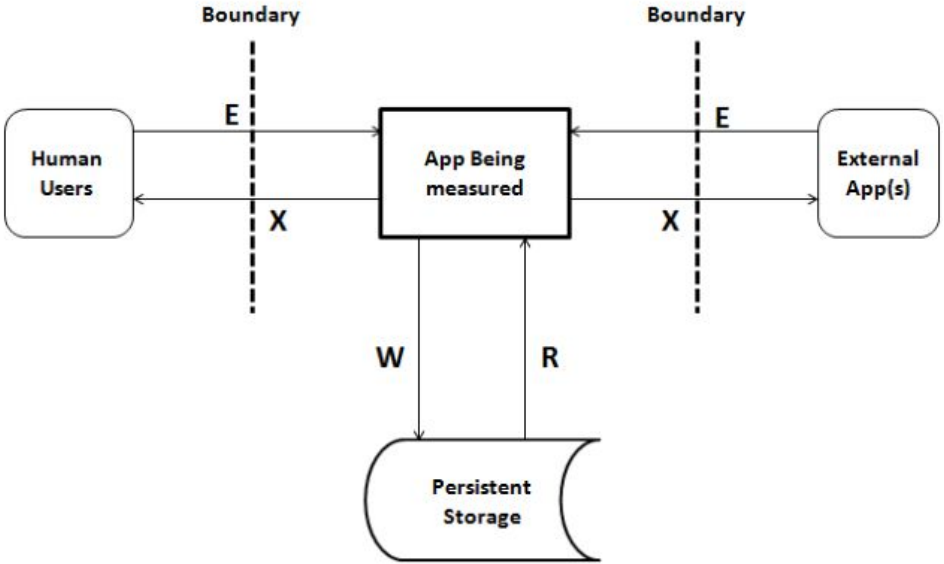
ARM Architecture

- 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**.



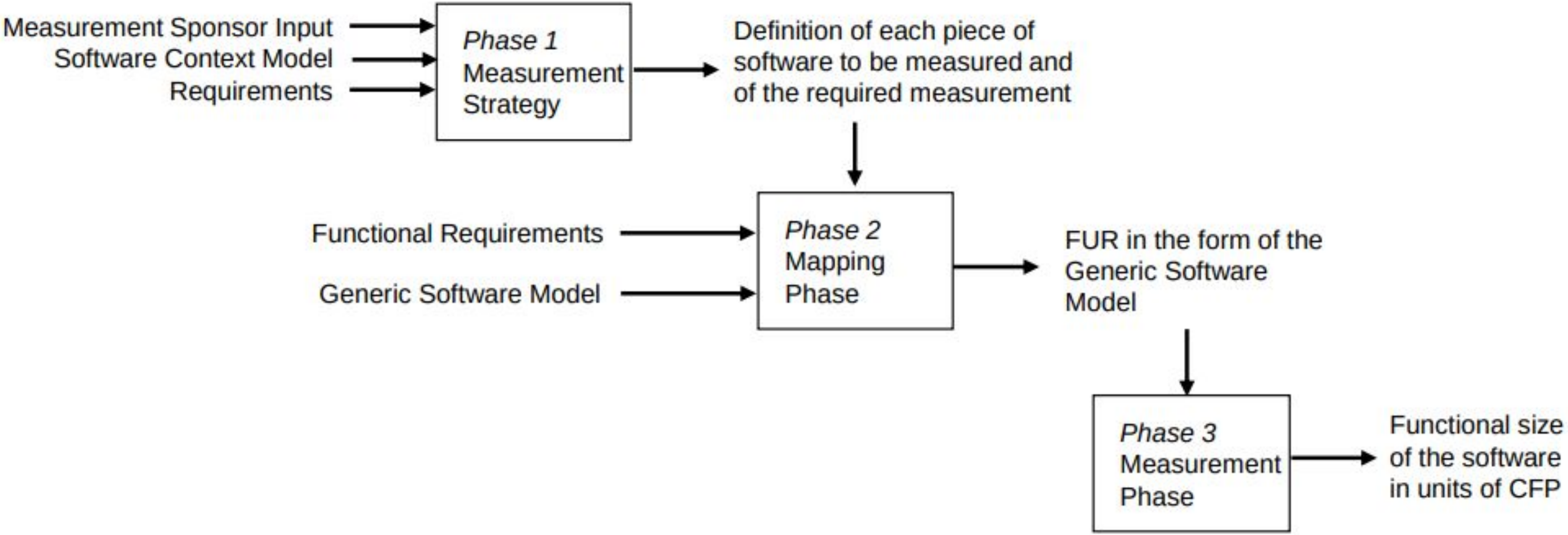
COSMIC Overview

- 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

Presentation Agenda

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

```
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)
        updateStatusRegsiter()

    return;
}
```

The ADC Example

```
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

```
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;
}
```

Presentation Agenda

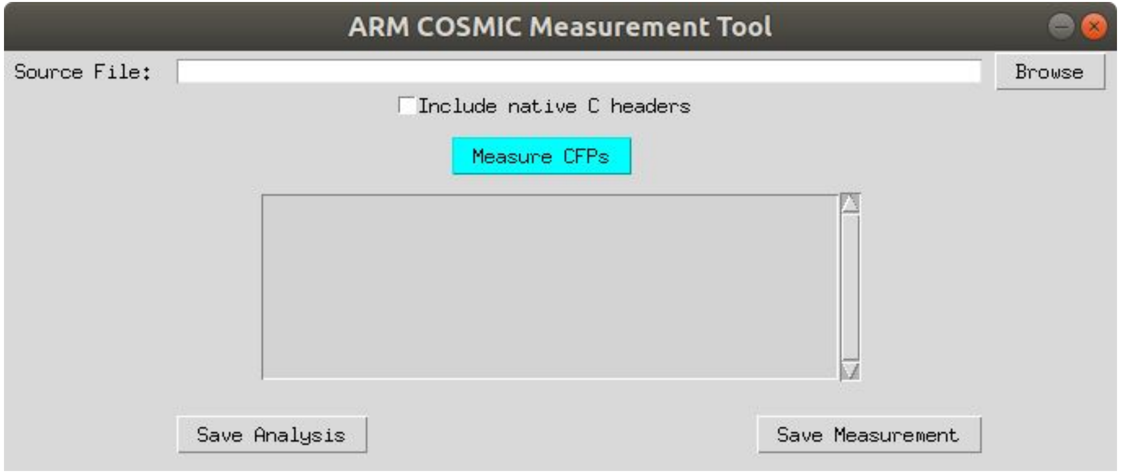
In this presentation, we:

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**.

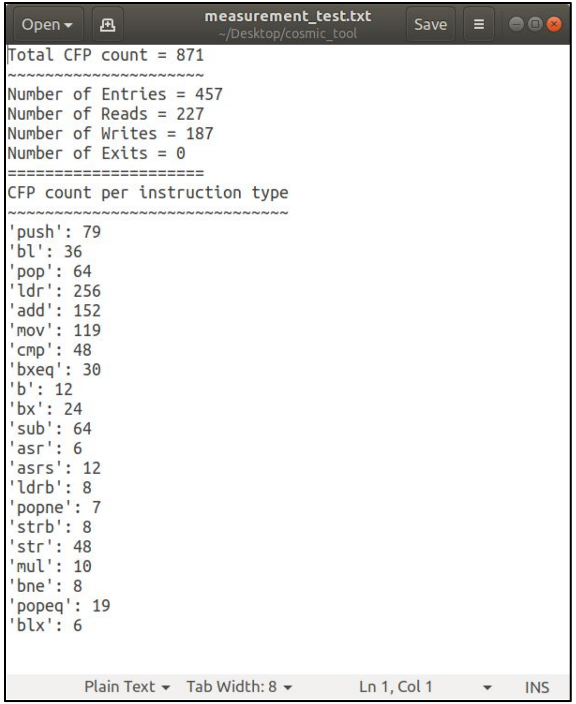
Automated Measurement Tool Prototype

- 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.

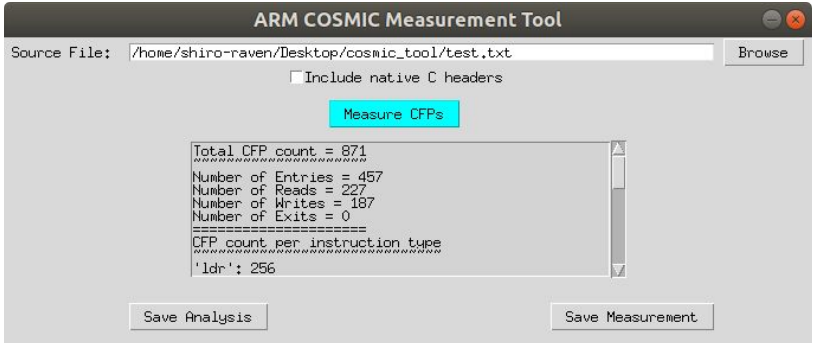
```
00010460 <factorial>:
   10460:  e92d4800   push {fp, lr}
   10464:  e28db004   add  fp, sp, #4
   10468:  e24dd008   sub  sp, sp, #8
   1046c:  e50b0008   str  r0, [fp, #-8]
   10470:  e51b3008   ldr  r3, [fp, #-8]
   10474:  e3530000   cmp  r3, #0
   10478:  1a000001   bne  10484 <factorial+0x24>
   1047c:  e3a03001   mov  r3, #1
   10480:  ea000006   b    104a0 <factorial+0x40>
   10484:  e51b3008   ldr  r3, [fp, #-8]
   10488:  e2433001   sub  r3, r3, #1
   1048c:  e1a00003   mov  r0, r3
   10490:  ebfffff2   bl   10460 <factorial>
   10494:  e1a02000   mov  r2, r0
   10498:  e51b3008   ldr  r3, [fp, #-8]
   1049c:  e0030293   mul  r3, r3, r2
   104a0:  e1a00003   mov  r0, r3
   104a4:  e24bd004   sub  sp, fp, #4
   104a8:  e8bd8800   pop  {fp, pc}

000104ac <main>:
   104ac:  e92d4800   push {fp, lr}
   104b0:  e28db004   add  fp, sp, #4
```

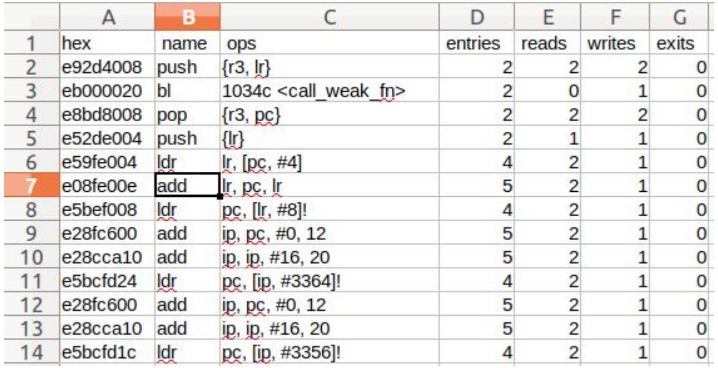


Automated Measurement Tool Prototype




```
Total CFP count = 871
~~~~~~~~~~~~~~~~~~~~~
Number of Entries = 457
Number of Reads = 227
Number of Writes = 187
Number of Exits = 0
=====================
CFP count per instruction type
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'push': 79
'bl': 36
'pop': 64
'ldr': 256
'add': 152
'mov': 119
'cmp': 48
'bxeq': 30
'b': 12
'bx': 24
'sub': 64
'asr': 6
'asrs': 12
'ldrb': 8
'popne': 7
'strb': 8
'str': 48
'mul': 10
'bne': 8
'popeq': 19
'blx': 6
```





	A	B	C	D	E	F	G
1	hex	name	ops	entries	reads	writes	exits
2	e92d4008	push	{r3, lr}	2	2	2	0
3	eb000020	bl	1034c <call_weak_fn>	2	0	1	0
4	e8bd8008	pop	{r3, pc}	2	2	2	0
5	e52de004	push	{lr}	2	1	1	0
6	e59fe004	ldr	lr, [pc, #4]	4	2	1	0
7	e08fe00e	add	lr, pc, lr	5	2	1	0
8	e5bef008	ldr	pc, [lr, #8]!	4	2	1	0
9	e28fc600	add	ip, pc, #0, 12	5	2	1	0
10	e28cca10	add	ip, ip, #16, 20	5	2	1	0
11	e5bcfd24	ldr	pc, [ip, #3364]!	4	2	1	0
12	e28fc600	add	ip, pc, #0, 12	5	2	1	0
13	e28cca10	add	ip, ip, #16, 20	5	2	1	0
14	e5bcfd1c	ldr	pc, [ip, #3356]!	4	2	1	0

Presentation Agenda

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