TOWARDS UNIVERSAL COSMIC SIZE MEASUREMENT AUTOMATION

H. Soubra, Y. Abufrikha et A. Abran
INTRODUCTION

• There are 8945 different programming languages!

• Automating COSMIC-based FSM: complete mapping between the principles of the COSMIC method and the notation of a programming language.

• And different programming languages may produce different results when implementing the same set of requirements.

• This is a feasibility study of an approach to a ‘universal’ tool based on COSMIC ISO 19761 and MIPS to automate the measurement of software written in different programming languages.
AGENDA

- Introduction
- COSMIC Overview
- Related Work
- Proposed Approach
  - MIPS Overview
  - Mapping COSMIC to MIPS
  - Automation Prototype
- Conclusion
The COSMIC method measurement process.
(source: Measurement Manual)
AGENDA

• Introduction
• COSMIC Overview
• Related Work
• Proposed Approach
  • MIPS Overview
  • Mapping COSMIC to MIPS
  • Automation Prototype
• Conclusion
RELATED WORK

• Examples of COSMIC-based automation tools from the literature:

<table>
<thead>
<tr>
<th>Tool</th>
<th>Year</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool using Simulink model</td>
<td>2011</td>
<td>Soubra et al. [4]</td>
</tr>
<tr>
<td>ScopeMaster</td>
<td>2018</td>
<td>Hammond et al. [5]</td>
</tr>
<tr>
<td>A tool for the automation of functional size measurement with RUP</td>
<td>2004</td>
<td>Azzouz and Abran [9]</td>
</tr>
<tr>
<td>μcROSE</td>
<td>2005</td>
<td>Diab et al. [10]</td>
</tr>
<tr>
<td>A procedure that measures the functional size</td>
<td>2015</td>
<td>Gomitas and Tarhan [11]</td>
</tr>
<tr>
<td>A tool using SCADE model</td>
<td>2015</td>
<td>Soubra et al. [12]</td>
</tr>
<tr>
<td>UML profile tool</td>
<td>2011</td>
<td>Lind and Heldal [13]</td>
</tr>
</tbody>
</table>

• None of the proposed tools can be considered universal since they require specific types of input languages/models.

• The concept of a universal tool is a tool that is applicable to all types of input languages/models.
AGENDA

- Introduction
- COSMIC Overview
- Related Work
- Proposed Approach
  - MIPS Overview
  - Mapping COSMIC to MIPS
  - Automation Prototype
- Conclusion
Once a program is translated into machine code, it becomes independent of the original language it was written in.

Map and use machine code as Input for COSMIC-FSM

Use MIPS as POC and generalize approach

PROPOSED APPROACH: MIPS OVERVIEW

• MIPS-Microprocessor without Interlocked Pipelined Stages: is a reduced instruction set computer (RISC)

• Widely used in Computer Architecture courses and has been used in some implementations in game consoles: Nintendo 64 and PlayStation.
PROPOSED APPROACH: MIPS OVERVIEW

- Five-stage execution pipeline: fetch, decode, execute, memory-access, write-result.
- Regular instruction set, all instructions are 32-bit.
- Three-operand arithmetical and logical instructions.
- 32 general-purpose registers of 32-bits each.
- …

- The components of MIPS architecture are:
  - MIPS instruction set architecture (ISA)
  - MIPS privileged resource architecture (PRA)
  - MIPS modules and application-specific extensions (ASEs)
  - MIPS user defined instructions (UDIs)

...
ISA AND MACHINE CODE

- Two sides of the same coin!
- ISA serves as the boundary between the software and hardware.

MIPS code example (file)

```
addi $s6, $0, 23
addi $t5, $0, 5
sw $t5, 0($s6)
addi $s6, $0, 8
Loop:
sll $t1, $s3, 2
add $t1, $t1, $s6
lw $t0, 0($t1)
bne $t0, $zero, Exit
addi $s3, $s3, 1
j Loop
Exit:
```

add $s1,$s2,$s3
MIPS uses **byte addressing** to access memory operands

Objects must be **aligned**

- MIPS addressing modes are **Register**, **Immediate**, and **Displacement** [where a constant offset is added to a register to form the memory address].

```
add $t0,$s1,$s2
addi $s3,$s3,4
lb $t0,13($s2)
```

```
$\text{Base} \quad 4 \text{ bytes}
```

---

**ISA - ADDRESSING MODES AND MEMORY**

Soubra, Abufrikha, Abran ©2020
# ISA - OPERATIONS/CONTROL

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Meaning</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>Add</td>
<td>R</td>
</tr>
<tr>
<td>ADDI</td>
<td>Add Immediate</td>
<td>I</td>
</tr>
<tr>
<td>ADDIU</td>
<td>Add Unsigned Immediate</td>
<td>I</td>
</tr>
<tr>
<td>ADDU</td>
<td>Add Unsigned</td>
<td>R</td>
</tr>
<tr>
<td>AND</td>
<td>Bitwise AND</td>
<td>R</td>
</tr>
<tr>
<td>ANDI</td>
<td>Bitwise AND Immediate</td>
<td>I</td>
</tr>
<tr>
<td>BEQ</td>
<td>Branch if Equal</td>
<td>I</td>
</tr>
<tr>
<td>BLEZ</td>
<td>Branch if Less Than or Equal to Zero</td>
<td>I</td>
</tr>
<tr>
<td>BNE</td>
<td>Branch if Not Equal</td>
<td>I</td>
</tr>
<tr>
<td>BGTZ</td>
<td>Branch on Greater Than Zero</td>
<td>I</td>
</tr>
<tr>
<td>DIV</td>
<td>Divide</td>
<td>R</td>
</tr>
<tr>
<td>DIVU</td>
<td>Unsigned Divide</td>
<td>R</td>
</tr>
<tr>
<td>j</td>
<td>Jump to Address</td>
<td>J</td>
</tr>
<tr>
<td>JAL</td>
<td>Jump and Link</td>
<td>J</td>
</tr>
<tr>
<td>JR</td>
<td>Jump to Address in Register</td>
<td>R</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LB</td>
<td>Load Byte</td>
</tr>
<tr>
<td>LBU</td>
<td>Load Byte Unsigned</td>
</tr>
<tr>
<td>LHU</td>
<td>Load Halfword Unsigned</td>
</tr>
<tr>
<td>LUI</td>
<td>Load Upper Immediate</td>
</tr>
<tr>
<td>LW</td>
<td>Load Word</td>
</tr>
<tr>
<td>MFHI</td>
<td>Move from HI Register</td>
</tr>
<tr>
<td>MTHI</td>
<td>Move to HI Register</td>
</tr>
<tr>
<td>MFLO</td>
<td>Move from LO Register</td>
</tr>
<tr>
<td>MTLO</td>
<td>Move to LO Register</td>
</tr>
<tr>
<td>MFLO</td>
<td>Move from Coprocessor 0</td>
</tr>
<tr>
<td>MULT</td>
<td>Multiply</td>
</tr>
<tr>
<td>MULTU</td>
<td>Unsigned Multiply</td>
</tr>
<tr>
<td>NOR</td>
<td>Bitwise NOR (NOT-OR)</td>
</tr>
<tr>
<td>XOR</td>
<td>Bitwise XOR (Exclusive-OR)</td>
</tr>
<tr>
<td>OR</td>
<td>Bitwise OR</td>
</tr>
<tr>
<td>ORI</td>
<td>Bitwise OR Immediate</td>
</tr>
<tr>
<td>SB</td>
<td>Store Byte</td>
</tr>
<tr>
<td>SH</td>
<td>Store Halfword</td>
</tr>
<tr>
<td>SLT</td>
<td>Set to 1 if Less Than</td>
</tr>
<tr>
<td>SLTI</td>
<td>Set to 1 if Less Than Immediate</td>
</tr>
<tr>
<td>SLTIU</td>
<td>Set to 1 if Less Than Unsigned Immediate</td>
</tr>
<tr>
<td>SLTU</td>
<td>Set to 1 if Less Than Unsigned</td>
</tr>
<tr>
<td>SLL</td>
<td>Logical Shift Left</td>
</tr>
<tr>
<td>SRL</td>
<td>Logical Shift Right (0-extended)</td>
</tr>
<tr>
<td>SRA</td>
<td>Arithmetic Shift Right (sign-extended)</td>
</tr>
<tr>
<td>SUB</td>
<td>Subtract</td>
</tr>
<tr>
<td>SUBU</td>
<td>Unsigned Subtract</td>
</tr>
<tr>
<td>SW</td>
<td>Store Word</td>
</tr>
</tbody>
</table>

**Notes:**
- Memory operation expl
- Register operation expl

---

Soubra, Abufrikha, Abran ©2020
AGENDA

• Introduction
• COSMIC Overview
• Related Work
• Proposed Approach
  • MIPS Overview
  • Mapping COSMIC to MIPS
  • Automation Prototype
• Conclusion
# Mapping COSMIC to MIPS

<table>
<thead>
<tr>
<th>Rule number</th>
<th>COSMIC element</th>
<th>Rule description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Functional Process (FP)</td>
<td>Identify 1 functional process for each subroutine in the file.</td>
</tr>
<tr>
<td>2</td>
<td>Data movement</td>
<td>Identify 1 Entry (E) for each source register in each instruction in a FP.</td>
</tr>
<tr>
<td>3</td>
<td>Data movement</td>
<td>Identify 1 Entry (E) for each immediate each instruction in a FP.</td>
</tr>
<tr>
<td>4</td>
<td>Data movement</td>
<td>Identify 1 Exit (X) for a destination register in each instruction in a FP.</td>
</tr>
<tr>
<td>5</td>
<td>Data movement</td>
<td>Identify 1 Exit (X) for the new PC value after branch and jump instructions.</td>
</tr>
<tr>
<td>6</td>
<td>Data movement</td>
<td>Identify 1 Exit (X) for the return value after branch and link and jump and link instructions.</td>
</tr>
<tr>
<td>7</td>
<td>Data movement</td>
<td>Identify 1 Read (R) for each Load instruction inside a FP.</td>
</tr>
<tr>
<td>8</td>
<td>Data movement</td>
<td>Identify 1 Write (W) for each Store instruction inside a FP.</td>
</tr>
<tr>
<td>9</td>
<td>Functional process size</td>
<td>Aggregate the COSMIC Function Point (CFP) for each data movement in a FP, to obtain the size of the process.</td>
</tr>
<tr>
<td>10</td>
<td>Size of the software</td>
<td>Aggregate the CFP of each FP, to obtain the size of the whole software.</td>
</tr>
</tbody>
</table>
AGENDA

• Introduction
• COSMIC Overview
• Related Work
• Proposed Approach
  • MIPS Overview
  • Mapping COSMIC to MIPS
  • Automation Prototype
• Conclusion
**AUTOMATION PROTOTYPE**

- Parse **Input MIPS code file**
- **store it in an ArrayList**
- **perform string manipulation**
- **obtain the Opcode + labels (Subroutines)**

**Opcode** = MIPS operation code (Unique instruction ID)

- **Identify** E, X, R, W according to the rules
- **Yield measurement result**

**Proto Developed in JAVA**

COSMIC for MIPS
AUTOMATION PROTOTYPE

MIPS code Input file: TestFile.asm

addi $s6, $0, 23
addi $t5, $0, 5
sw $t5, 0 ( $s6 )
addi $s6, $0, 8
Loop:
sll $t1, $s3, 2
add $t1, $t1, $s6
lw $t0, 0 ( $t1 )
bne $t0, $zero, Exit
add $s3, $s3, 1
j Loop
Exit:

Output file:

This is the report corresponding to TestFile.asm
The file had 14 lines of code.
The total number of instructions in the file = 10 instruction(s).
COSMIC calculation:
The file had 3 functional processes.
Path
Loop
Exit
The functional size = 33 CFP.
The total number of entries = 22.
The instructions that caused the number of Entries:
add $s6, $0, 23
add $t5, $0, 5
sw $t5, 0 ( $s6 )
add $s6, $0, 8
Loop:
sll $t1, $s3, 2
add $t1, $t1, $s6
lw $t0, 0 ( $t1 )
bne $t0, $zero, Exit
add $s3, $s3, 1
j Loop
Exit:

The total number of exits = 9.
The instructions that caused the number of Exits:
add $s6, $0, 23
add $t5, $0, 5
sw $t5, 0 ( $s6 )
add $s6, $0, 8
Loop:
sll $t1, $s3, 2
add $t1, $t1, $s6
lw $t0, 0 ( $t1 )
bne $t0, $zero, Exit
add $s3, $s3, 1
j Loop
Exit:

The total number of read = 1.
The instructions that caused the number of Reads:
sw $t5, 0 ( $s6 )
add $s3, $s3, 1
lw $t0, 0 ( $t1 )
bne $t0, $zero, Exit
add $s3, $s3, 1
j Loop
Exit:

Output file (ctnd):
AUTOMATION PROTOTYPE

Limitations:

• This is a feasibility prototype
• Not all MIPS instructions -including pseudo-instructions- are included.
• Accuracy and the precision of the tool should be analyzed with more test cases.
AGENDA

- Introduction
- COSMIC Overview
- Related Work
- Proposed Approach
  - MIPS Overview
  - Mapping COSMIC to MIPS
  - Automation Prototype
- Conclusion
CONCLUSION

1. The goal of this study: to propose an approach for a `universal' tool based on COSMIC ISO 19761 to ensure that the measurement of all types of input software written in different programming languages is correctly automated.

2. Outcomes of the study: Mapping rules and a feasibility prototype tool based on COSMIC and MIPS was developed using Eclipse Java based on the latest version of the MIPS architecture.

3. Promising results: the 'universal' tool may be achieved by generalizing the approach proposed in this study to cover, and use, the machine code (ISA-Instruction Set Architecture) generated by any compiler/Assembler to get the COSMIC functional size of a program.
Q&A

hassan.soubra@guc.edu.eg
yomna.abufrihka@student.guc.edu.eg
alain.abran@etsmtl.ca