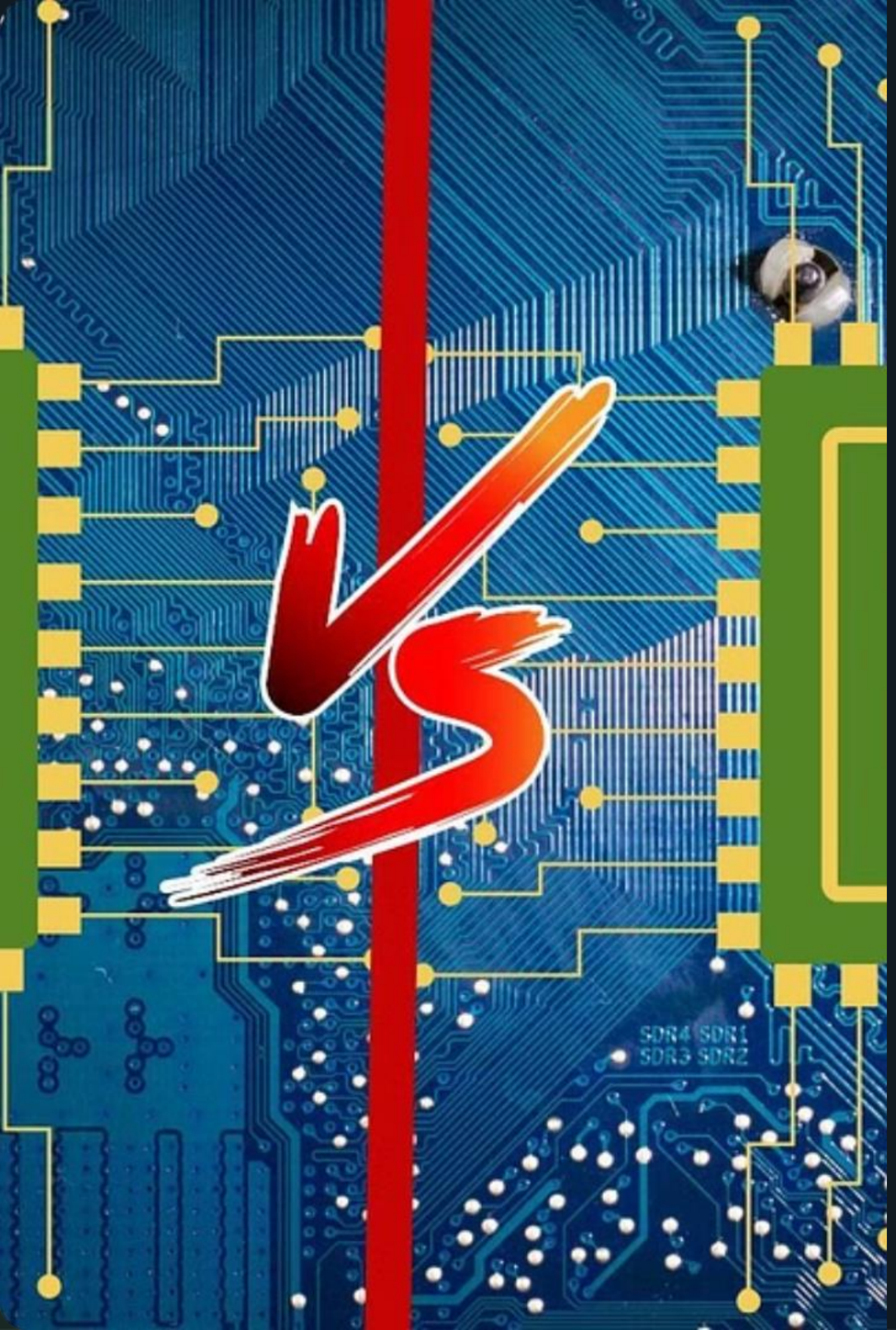


ARM Assembly Generation for a graduate compiler course

This presentation explores the process of ARM assembly generation within the context of a graduate compiler course.

- A.Jayasimha (192211992)
- A.Sivaprasad(192210398)
- S.Ragunatha (192211940)

IMPORTANCE :

- Educational Value
- Skill Development
- Research and Development
- Assessment and Improvement
- Practical Application

The importance of evaluating ARM assembly generation in a graduate compiler course extends beyond mere academic exercise. It equips students with crucial skills, fosters a deeper understanding of computer architecture and compiler design, and prepares them for both industry roles and advanced research in computer science.

INTRODUCTION:

- An important step in learning how software works and interacts with the hardware of a computer is learning about the compiler
- Our paper compares building an assembly generator for a Raspberry Pi using ARM to a previously built x86 assembly generator, to evaluate which works better for teaching purposes
- ARM & RISC ARM computers are a type of RISC computers

Overview of Compiler Architecture

1. **Lexical Analysis**

   The first phase of compilation, where the source code is tokenized.

2. **Syntax Analysis**

   This phase parses the input stream into a hierarchical structure.

3. **Semantic Analysis**

   Semantic analysis ensures that the source code conforms to the language's rules.

4. **Intermediate Code Generation**

   Produces an intermediate representation independent of the target architecture.

5. **Code Optimization**

   Optimizes the intermediate code to enhance performance.

6. **Code Generation**

   Translates the optimized code into the target assembly language.

ARM Instruction Set and Addressing Modes

Data Processing Instructions

Perform arithmetic and logical operations on data.

- ADD
- SUB
- MUL
- AND
- OR

Data Transfer Instructions

Move data between registers and memory.

- LDR
- STR
- MOV

Addressing Modes

Specify how operands are accessed in memory.

- Register Direct
- Immediate
- Register Indirect

Register Allocation and Spilling

Register Allocation

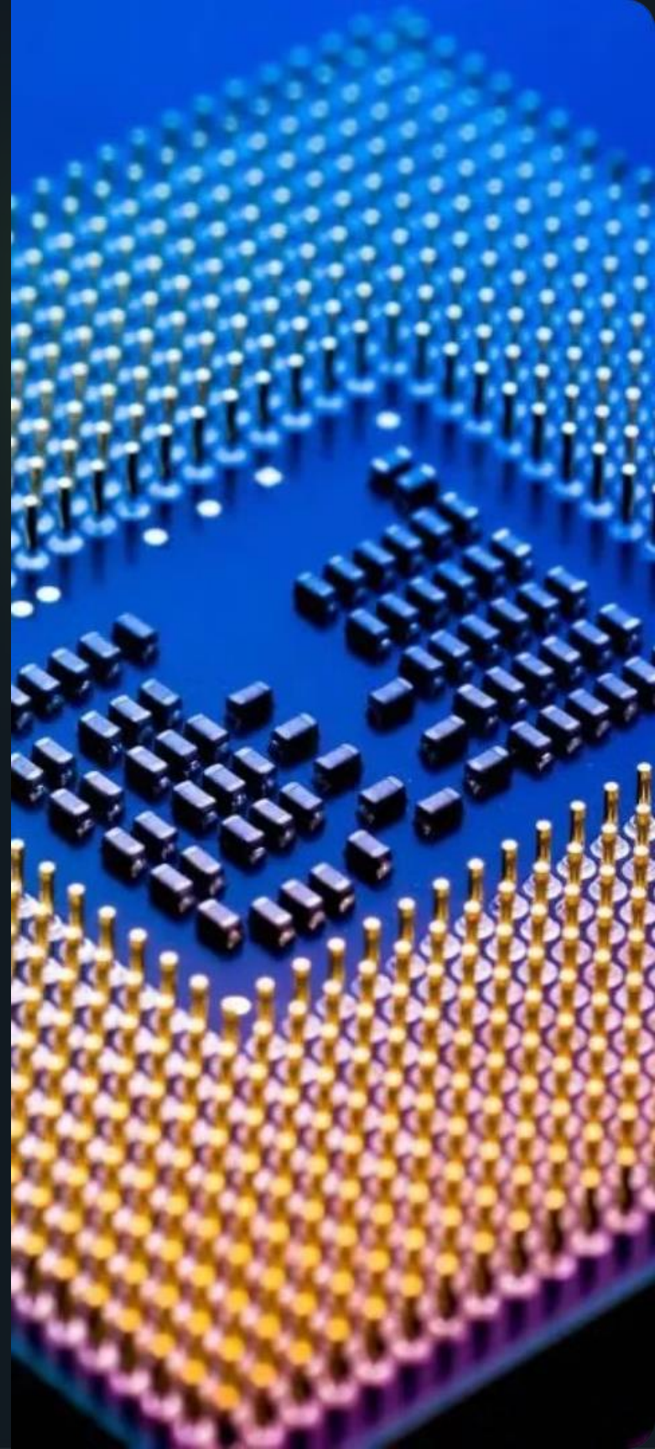
Assigns variables to registers to minimize memory access.

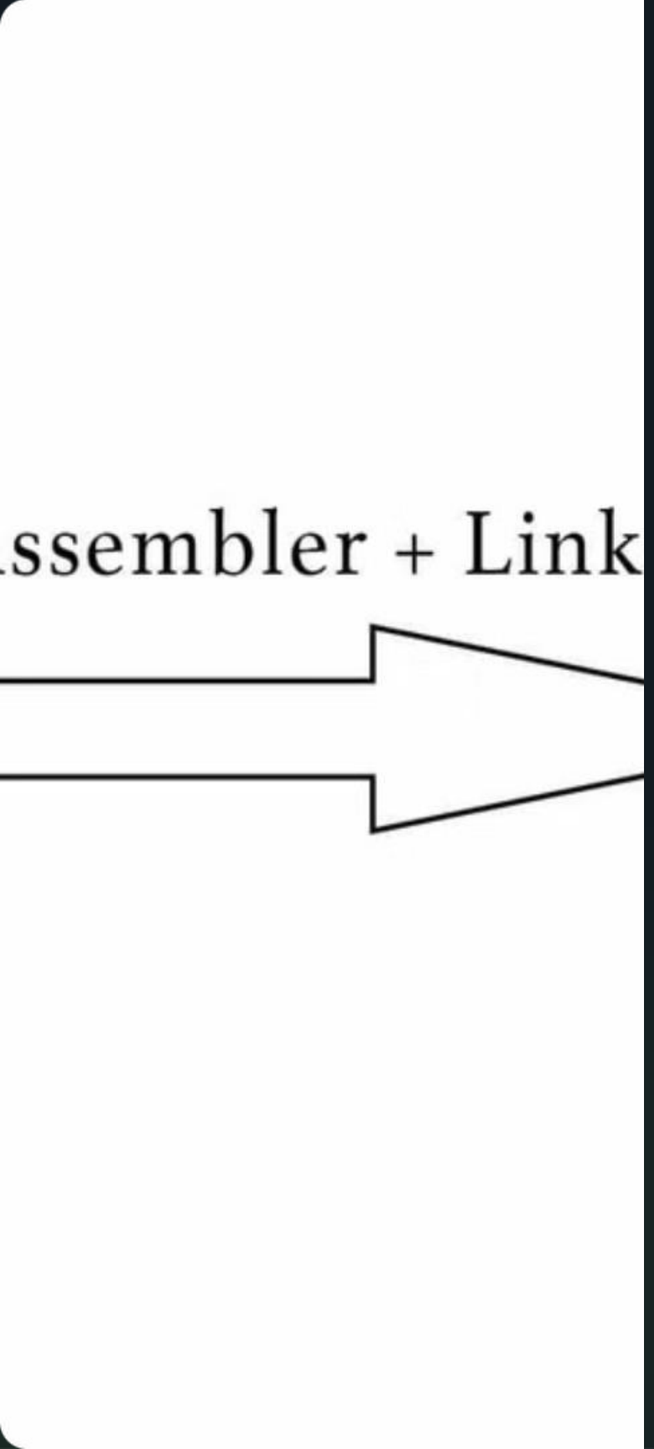
Register Spilling

When there are more variables than available registers, variables are temporarily stored in memory.

Graph Coloring Algorithm

A common technique used for register allocation, based on graph theory.





Instruction Selection and Code Generation

1

Intermediate Representation

A language-independent representation of the program.

2

Instruction Selection

Chooses the most efficient ARM instructions for each operation.

3

Code Generation

Generates the final ARM assembly code.

Handling Control Flow and Jumps

1

Conditional Jumps

Instructions that transfer control to a different part of the program based on conditions.

2

Unconditional Jumps

Instructions that transfer control to a different part of the program unconditionally.

3

Branch Prediction

A technique used to optimize control flow by predicting the target of a branch.

4

Loop Unrolling

A code optimization technique that expands loops to reduce the number of conditional jumps.

Optimizations for ARM Assembly

Optimization	Description
Common Subexpression Elimination	Identifies and eliminates redundant computations.
Loop Invariant Code Motion	Moves computations outside of loops if their values remain constant.
Strength Reduction	Replaces expensive operations with less expensive ones.

Evaluation and Conclusion

Performance

Evaluate the generated assembly code's execution speed and efficiency.

Code Size

Assess the generated assembly code's size and memory footprint.

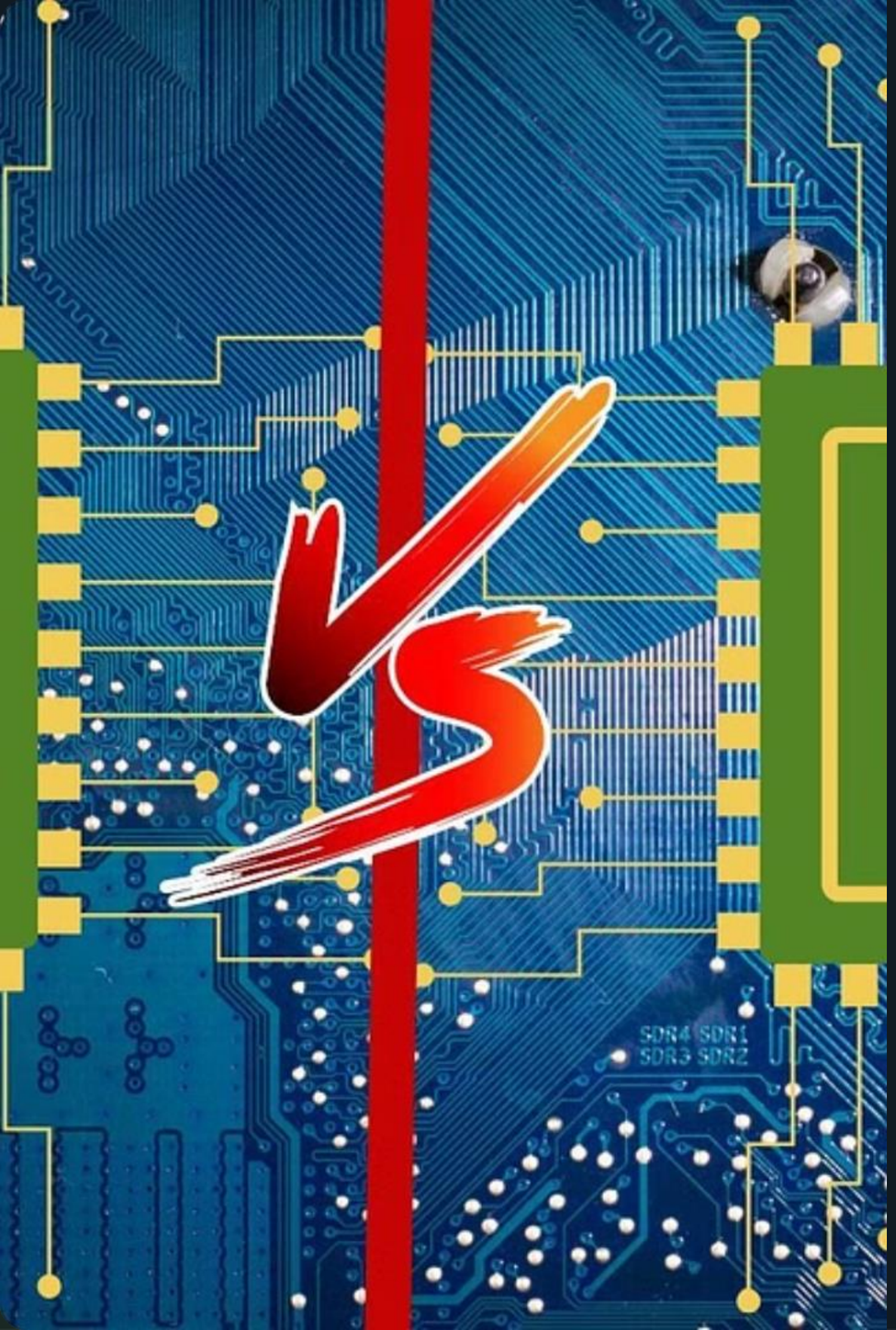
Correctness

Verify that the generated assembly code executes correctly and produces the expected results.

It equips students with crucial skills, fosters a deeper understanding of computer architecture and compiler design, and prepares them for both industry roles and advanced research in computer science.

Importance

Understanding ARM assembly generation is crucial for graduate compiler courses, enabling students to design and implement efficient compilers for ARM-based systems.



THANK YOU