SOFTWARE SYNTHESIS FOR EMBEDDED PROCESSORS

Lochana Abhayawardana

lochana.abhayawardana@stud.hshl.de

MOTIVATION

AGENDA

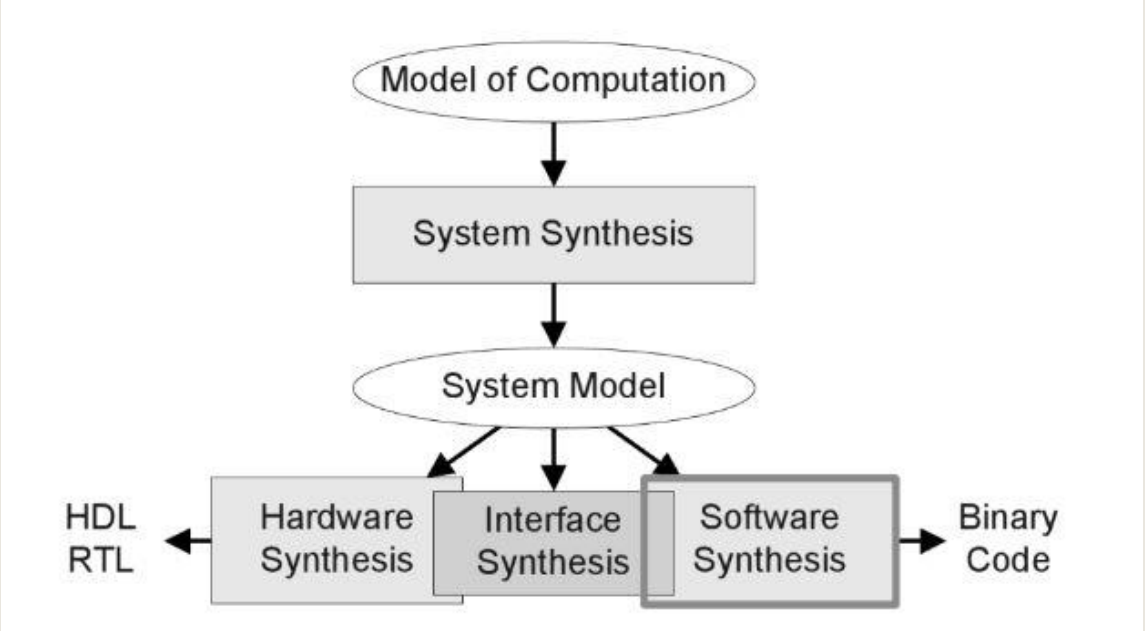
1. INTRODUCTION TO SOFTWARE SYNTHESIS FOR EMBEDDED SYSTEMS
2. APPROACHES TO SOFTWARE SYNTHESIS
3. PROGRAMMING IN HIGH LEVEL-LANGUAGES
4. COMPILERS
5. INSTRUCTION SET ARCHITECTURE (ISA)
6. Reduced Instruction Set Computer (RISC)
7. ARM CPU ARCHITECTURE
8. CONCLUSION

INTRODUCTION

- Embedded system in now days
- Software Synthesis – a critical role in reducing the design complexity
- Different approaches of the software synthesis
    1. High level synthesis
    2. Register Transfer level synthesis
    3. Instruction set synthesis

INTRODUCTION TO SOFTWARE SYNTHESIS FOR EMBEDDED SYSTEMS

- Software synthesis
- System synthesis
- Each part below System model uses it as an input and generates an implementation for each Hardware, interface and software synthesis.
- The Software Synthesis produces binary code for programmable processing elements.



APPROACHES TO SOFTWARE SYNTHESIS

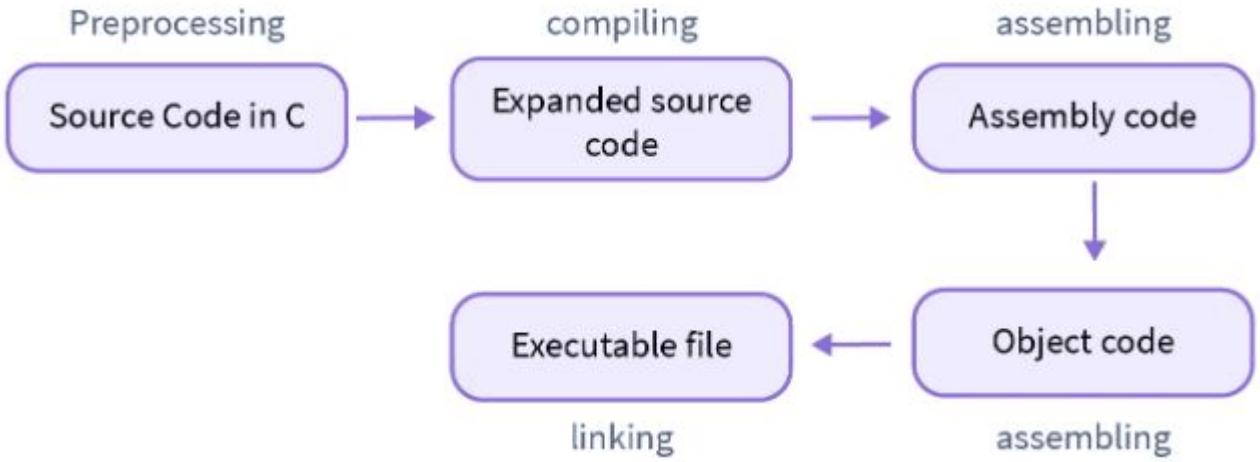
- Embedded systems : part of the physical control process.
- Embedded software can implement a control loop of the physical process through getting readings from the sensors.
- Therefore, the software is specific to the underlying hardware, in order to achieve that embedded systems use specialized components (hardware accelerators, processors, DPSs) with communication schemes.

PROGRAMMING IN HIGH LEVEL-LANGUAGES

- What is High level Languages:
- different from human languages
- How to write a program
- computers can only understand binary numbers.
- translators to lower the level of programming language

COMPILERS

- The high-level languages must be translated into low-level
- The compiler create an executable binary directly
- Assembly code .S
- Linker operations
- Libraries (.a, .dill) [2]



[2]

PRE-PROCESSING

- Pre-processing is the first step in the compilation process in C
- It is performed by the pre-processor tool
- All the statements with the symbol of #

- Steps,

1. Comments removal :

```c
/******************************************************************************

                            Online C Compiler.
                Code, Compile, Run and Debug C program online.
Write your code in this editor and press "Run" button to compile and execute it.

*******************************************************************************/

#include <stdio.h>
#define GRAV 9.8
int main()
{
    int a=4;  //accleraration
    int v=43; //velocity
    printf("Value of the Gravity is: %f ",GRAV);

    if (v>a){
        printf("\n%d",v);
    }

    return 0;
}

```

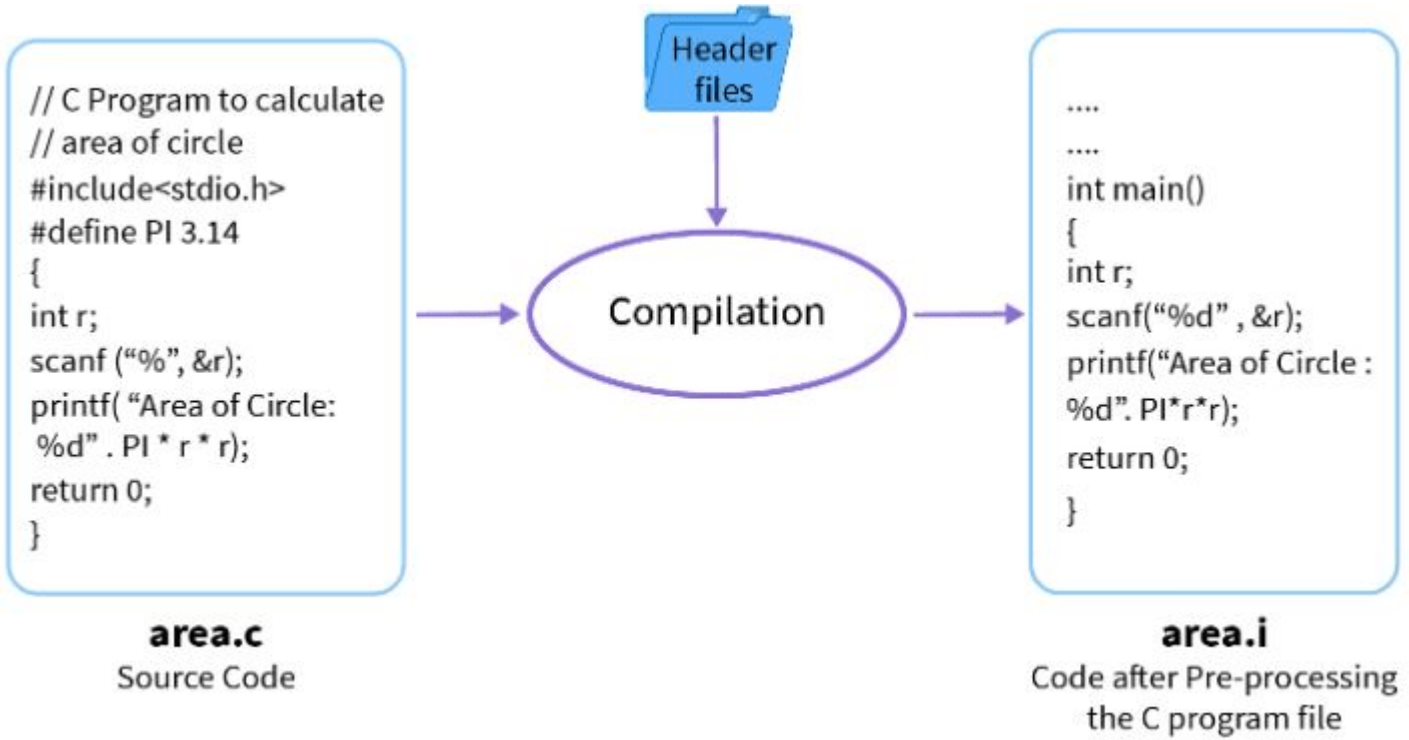
2. Macros expansion:

```c

#include <stdio.h>
#define GRAV 9.8
int main()
{
    int a=4;
    int v=43;
    printf("Value of the Gravity is: %f ",GRAV);

    if (v>a){
        printf("\n%d",v);
    }

    return 0;
}

```

3. File inclusion:

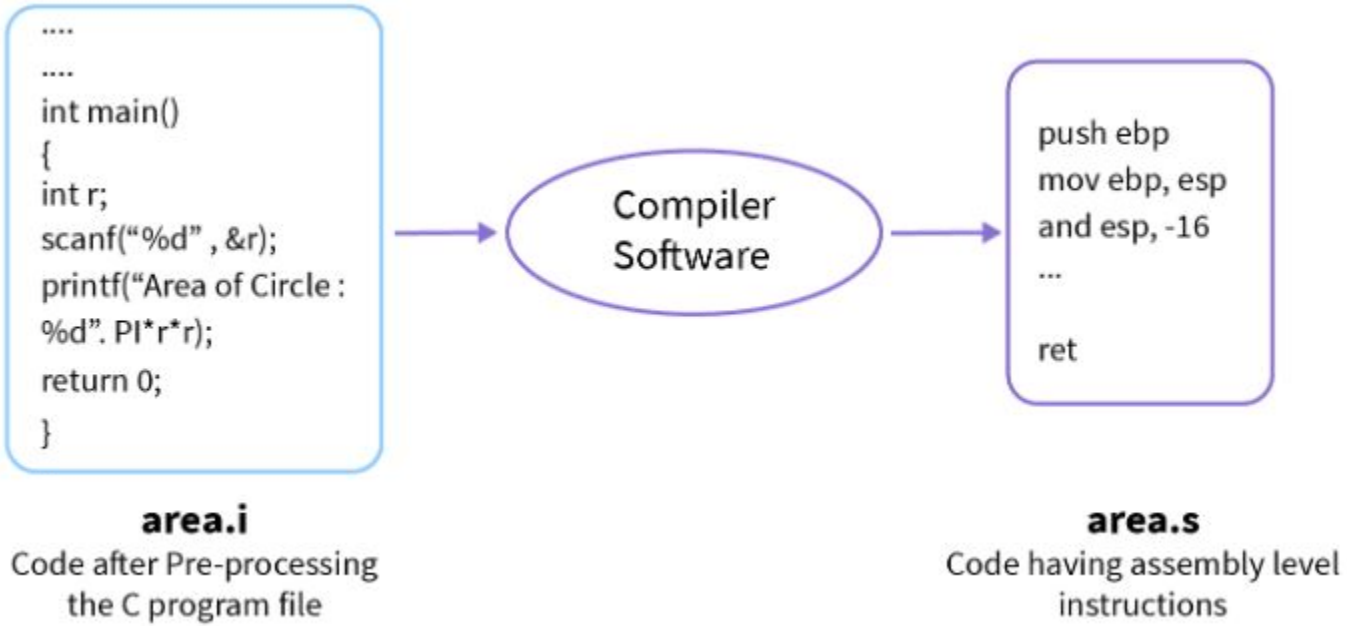
```c

#include <stdio.h>
int main()
{
    int a=4;
    int v=43;
    int GRAV= 9.8;
    printf("Value of the Gravity is: %f ",GRAV);

    if (v>a){
        printf("\n%d",v);
    }

    return 0;
}

```

3. Conditional compilation:



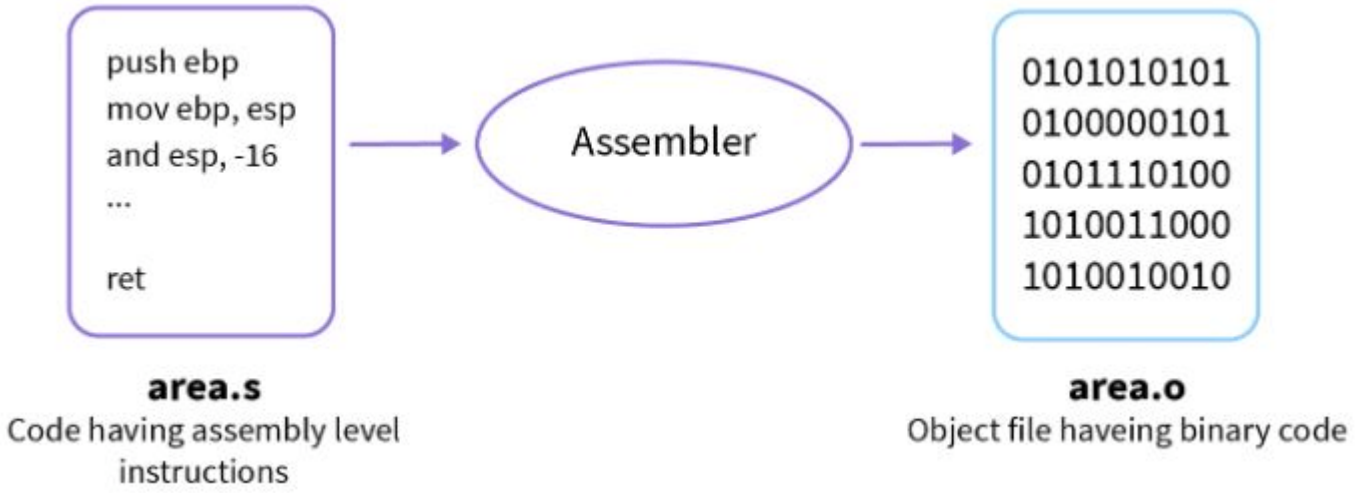
[2]

5. Compiling:



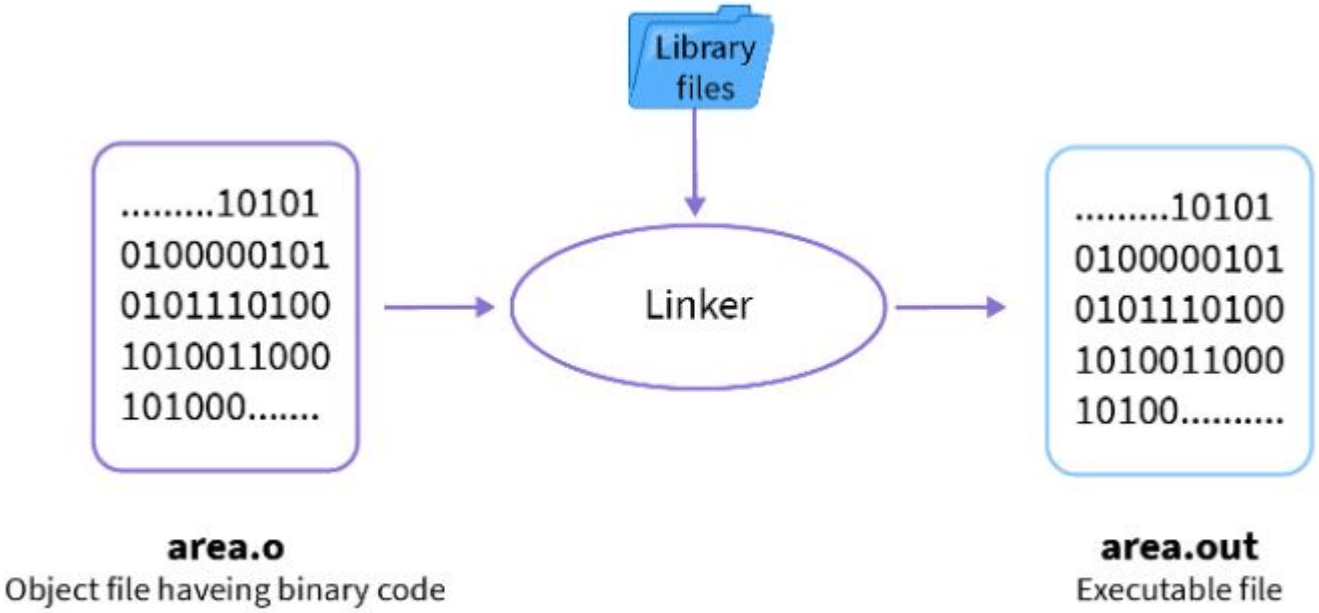
[2]

6. Assembling:



[2]

7. Linking


Library files
.....10101
0100000101
0101110100
1010011000
101000.....
Linker
.....10101
0100000101
0101110100
1010011000
10100.....
area.o
Object file haveing binary code
area.out
Executable file

[2]

INSTRUCTION SET ARCHITECTURE (ISA)

- produce a way to connect software to hardware.
- defines the type of instructions to be followed by the processor

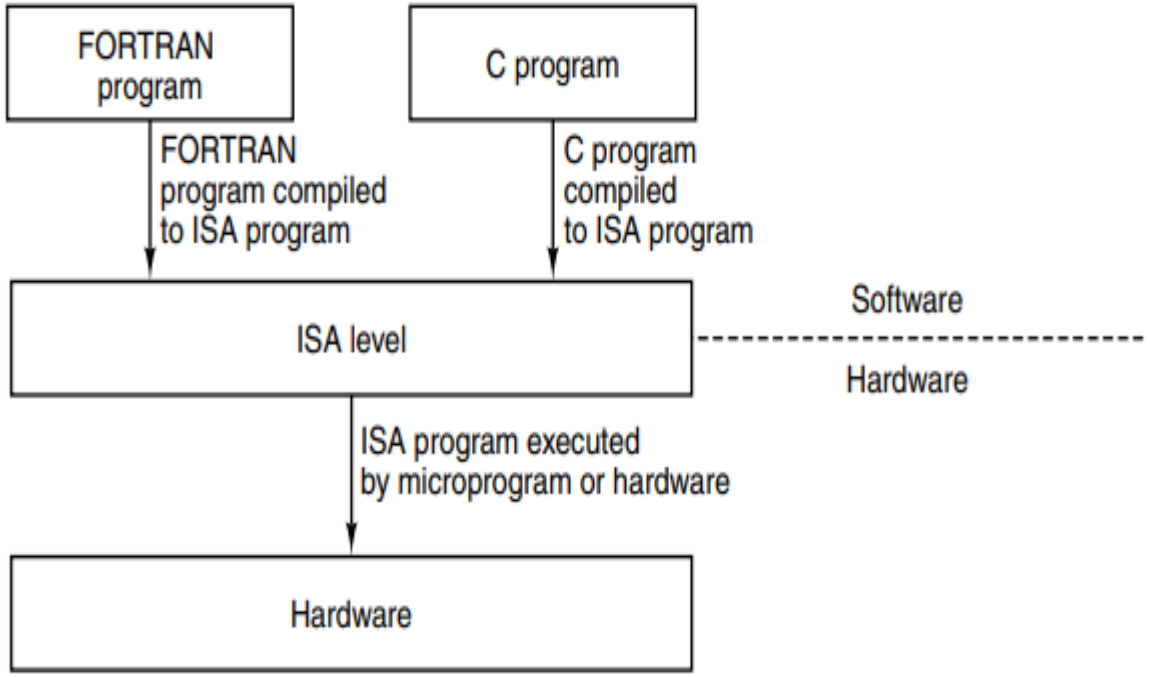
- Type of operations

1. Arithmetic /logic instructions:

2. Data Transfer instructions:

3. Branch and jump instructions:

ISA


FORTRAN program
C program
FORTRAN program compiled to ISA program
C program compiled to ISA program
ISA level
Software
Hardware
ISA program executed by microprogram or hardware
Hardware

REDUCED INSTRUCTION SET COMPUTER (RISC)

- At the start of RISC, the people who were working on microprogramming tools began to rethink the architectural design principles trying to close the “semantic gap” in high-level programming languages and microinstructions.
- optimizing compilers that can be used to compile high-level languages into instructions.
- These instructions are simpler

RISC

- Key principles of RISC

1) Functions should be simple.
2) Microinstructions should not be faster than simple instructions,
3) Moving software into microcode does not improve its performance or functionality
4) The RISC architecture prioritizes simple instruction decoding and pipeline execution Use of compiler technology to simplify instructions.

ARM CPU ARCHITECTURE

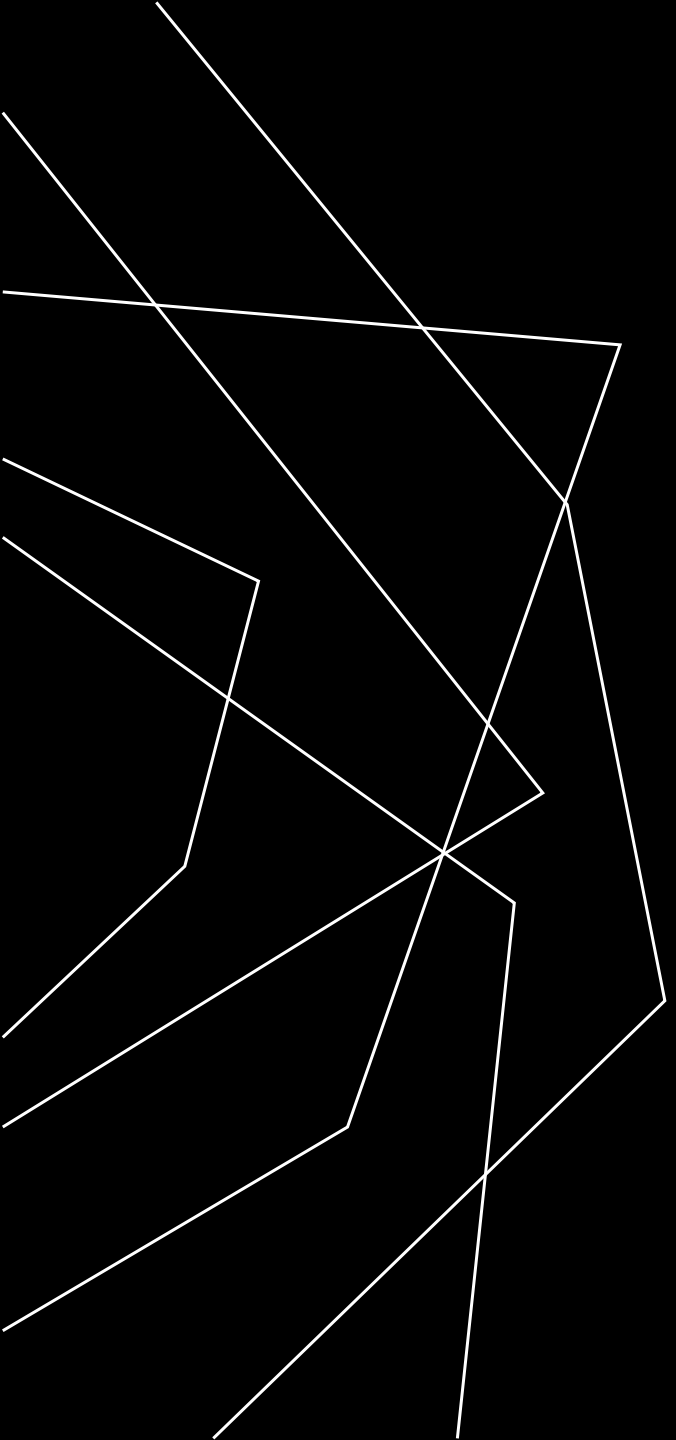
- ARM is a RISC architecture.
- The ARM ISA.
- All ARM instructions are 32-bit long
- three-operand encoding.
- large register file with 16 general-purpose registers, allowing pipelining

A. Registers:

- 16 general-purpose registers in the user mode.
- Register 15
- Register 14 .
- Register 13

B. **ARM architecture exceptions**

CONCLUSION



THANK YOU

Lochana Abhayawardana

Lochana.abhayawardana@stud.hshl.de