Turning Control Flow Graphs into Callgraphs

Transformation of partitioned codes for execution in heterogeneous architectures

PABLO BARRIO
TOBIAS KENTER
CARLOS CARRERAS
CHRISTIAN PLESSL
ROBERTO SIERRA

Departamento de Ingeniería Electrónica
Universidad Politécnica de Madrid

ETSIT UPM

Outline

1. Heterogeneous High Performance Computing

2. Compilation toolchain

3. Code refactoring for execution in heterogeneous platforms

Outline

1. **Heterogeneous High Performance Computing**

2. Compilation toolchain

3. Code refactoring for execution in heterogeneous platforms

High Performance Computing & Embedded Systems

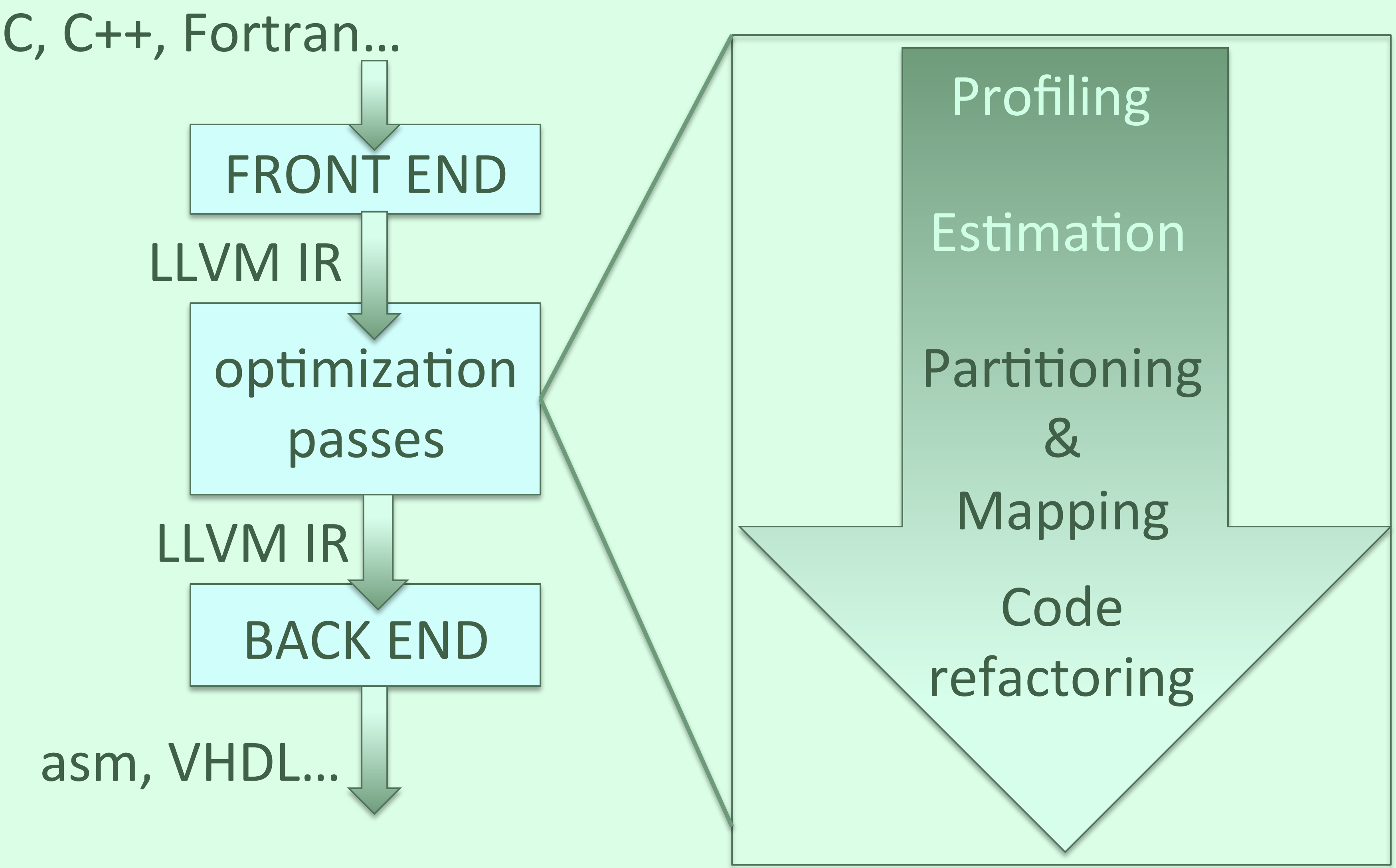
	Embedded	*HPC*
Type of processors	**Heterogeneous**	Homogeneous
Size	Small	**Massive**
Memory	Shared	**Distributed**

but getting closer every day…

Objectives

- A **code partitioner** for heterogeneous architectures.
- Easy to add **models for new devices and architectures**.
- Partitioning based on **software and hardware** characteristics.
- Communications generated for **distributed memory** systems.
- **Automatic parallelization**, both functional and data parallel.

The solution under research



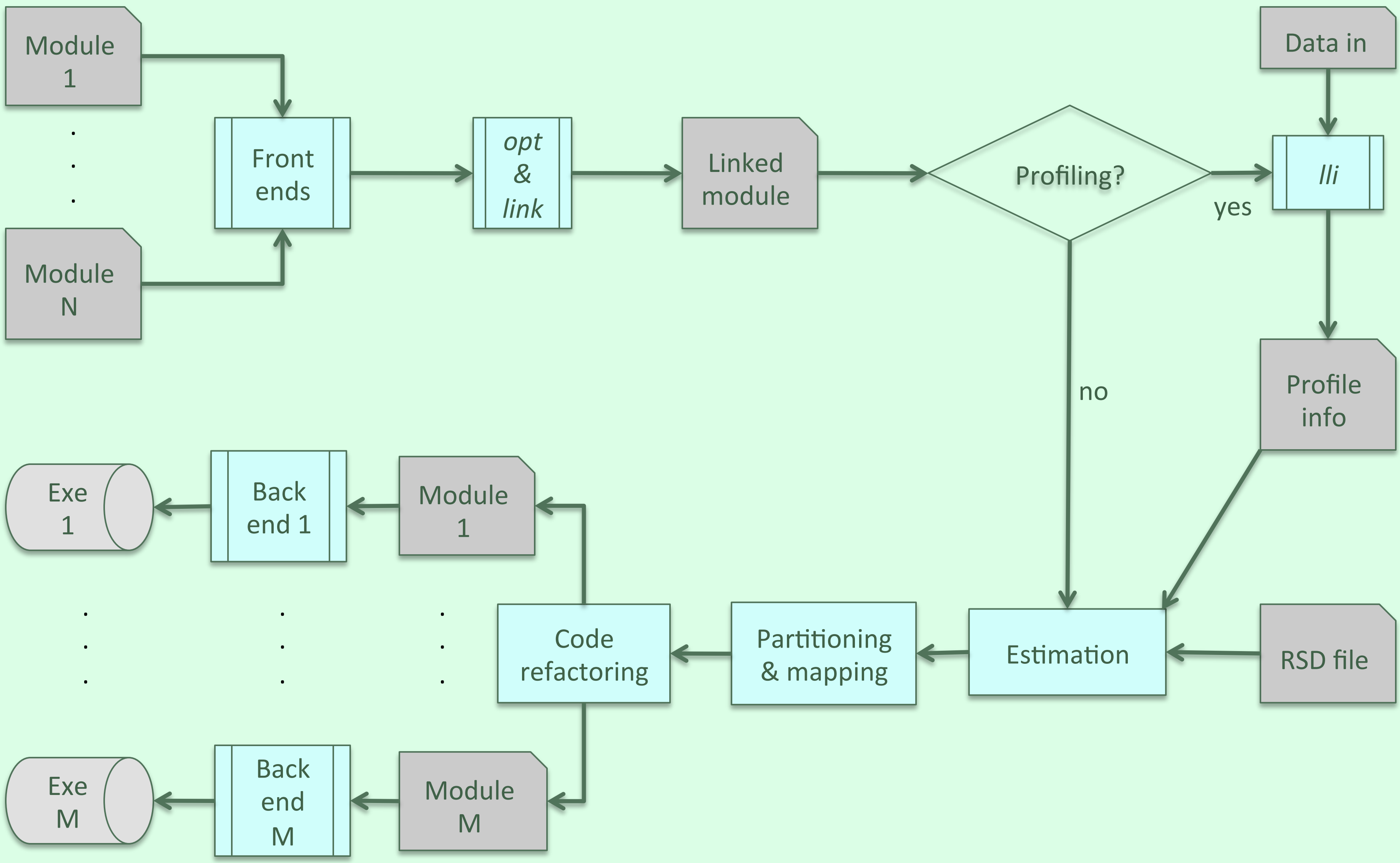
Outline

1. Heterogeneous High Performance Computing

2. Compilation toolchain

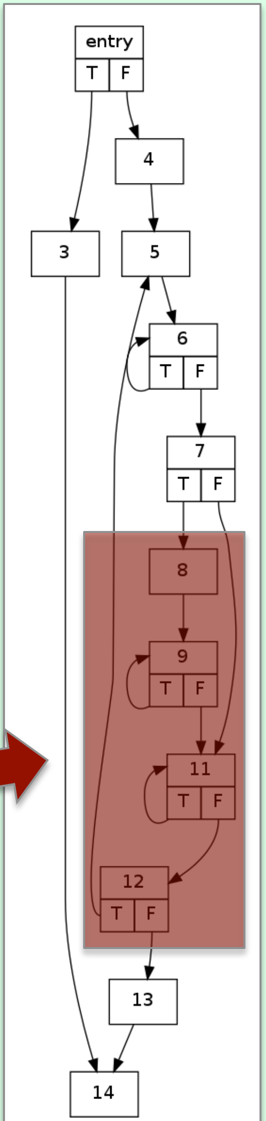
3. Code refactoring for execution in heterogeneous platforms

LLVM-based compilation toolchain



Partitioning & Mapping

```
[PartitioningPass] PARTITIONING OVERVIEW:
                 Initial exec time was 1.81e-07  s,
                 new is 1.06e-07
                 -- Speedup = 1.71e+00
[PartitionWriterPass] Generating partitioned code
PartitionWriterPass::runOnModule() -- Original
functions:
        odd with BBs:
                 entry --> CPU
        main with BBs:
                 entry --> CPU
                 3 --> CPU
                 4 --> CPU
                 beforeHeader --> CPU
                 5 --> CPU
                 6 --> CPU
                 7 --> CPU
                 8 --> CPU_SIMD
                 9 --> CPU_SIMD
                 11 --> CPU_SIMD
                 12 --> CPU_SIMD
                 13 --> CPU
                 14 --> CPU
                 afterHeader --> CPU
...
```



CFG for 'main' function

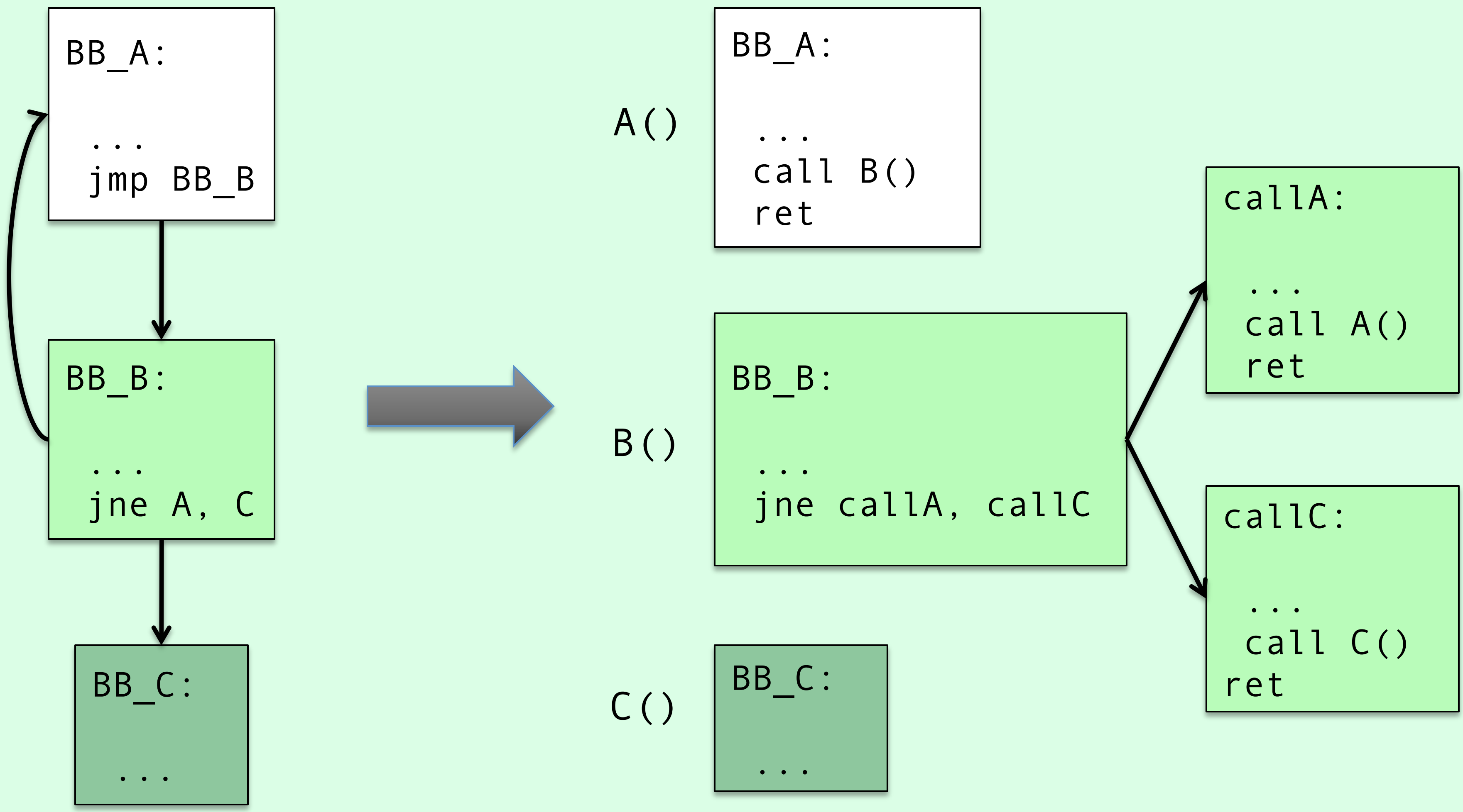
Outline

1. Heterogeneous High Performance Computing

2. Compilation toolchain

3. Code refactoring for execution in heterogeneous platforms

Function-based control flow



Refactoring methodology

duplicate constants

distribute globals

for every original function f

    initiatorList ← find initiators(f)

    create new functions(f, initiatorList)

    fix branches(initiatorList)

    fix phi nodes(initiatorList)

Refactoring methodology

duplicate constants

distribute globals

for every original function f

initiatorList ← find initiators(f)

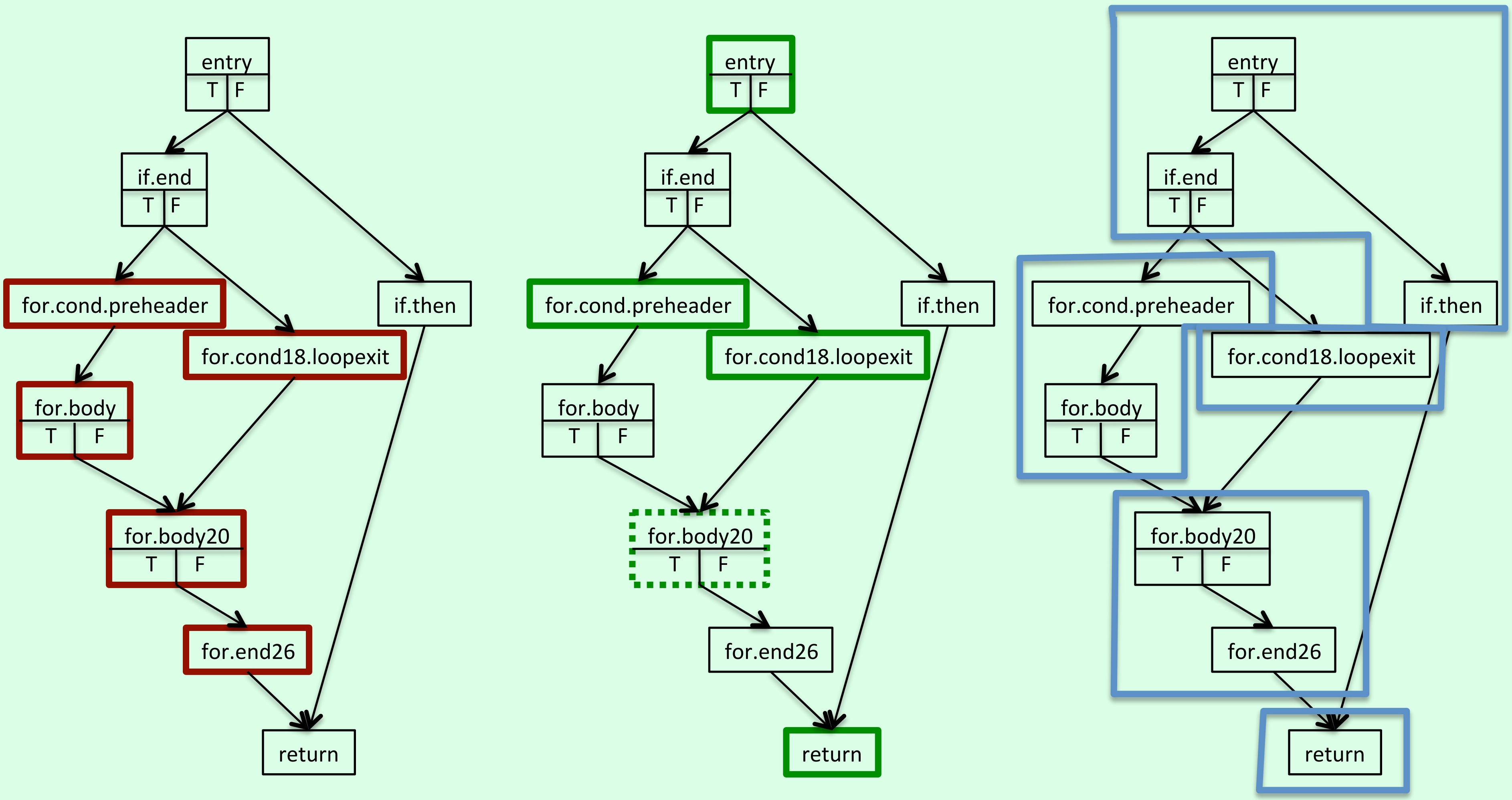
create new functions(f, initiatorList)

fix branches(initiatorList)

fix phi nodes(initiatorList)

Initiator list ← find initiators(f)

Partitioning result | Initiators | Resulting functions



Refactoring methodology

duplicate constants

distribute globals

for every original function f

    initiatorList ← find initiators(f)

    create new functions(f, initiatorList)

    fix branches(initiatorList)

    fix phi nodes(initiatorList)

create new functions (f, initiatorList)

MODULE 1

MODULE 2

```
i32 f(i8* %num)
```

```
BB_A:

 %3 = add i32 %2, 6
 jmp BB_B
```

DEV 1

```
BB_B:

 %4 = mul i32 %3, %3
 jne BB_A, BB_C
```

DEV 2

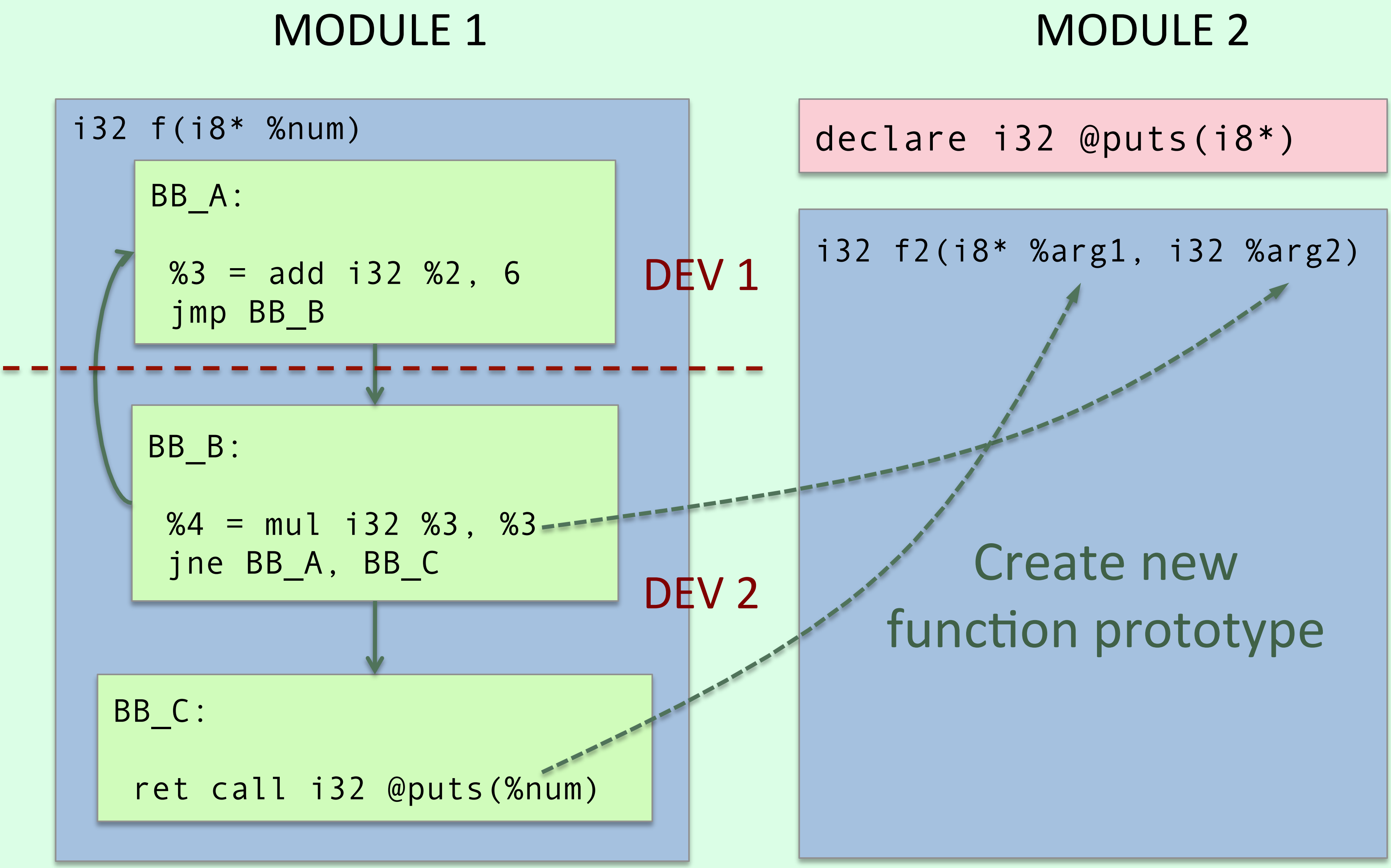
```
BB_C:

 ret call i32 @puts(%num)
```

```
declare i32 @puts(i8*)
```

Declare used functions in the destination module

Departamento de Ingeniería Electrónica
Universidad Politécnica de Madrid

Splitting functions



create new functions (f, initiatorList)

MODULE 1

```
i32 f(i8* %num)

BB_A:

 %3 = add i32 %2, 6
 jmp BB_B
```

Move Basic Blocks

MODULE 2

```
declare i32 @puts(i8*)
```

```
i32 f2(i8* %arg1, i32 %arg2)

BB_B:

 %4 = mul i32 %3, %3
 jne BB_A, BB_C

BB_C:

 ret call i32 @puts(%num)
```

create new functions (f, initiatorList)

MODULE 1

```
i32 f(i8* %num)

BB_A:

 %3 = add i32 %2, 6
 jmp BB_B
```

Fix argument uses

MODULE 2

```
declare i32 @puts(i8*)
```

```
i32 f2(i8* %arg1, i32 %arg2)

BB_B:

 %4 = mul i32 %arg2, %arg2
 jne BB_A, BB_C

BB_C:

 ret call i32 @puts(%arg1)
```

Refactoring methodology

duplicate constants

distribute globals

for every original function f

initiatorList ← find initiators(f)

create new functions(f, initiatorList)

fix branches(initiatorList)

fix phi nodes(initiatorList)

Departamento de
Ingeniería
Electrónica
Universidad Politécnica de Madrid

fix branches (initiatorList)

MODULE 1

```
i32 f(i8* %num)

BB_A:

 %3 = add i32 %2, 6
 %r = call i32 f2(%num, %3)
 ret %r
```

Replace old branches by function calls

MODULE 2

```
declare i32 @puts(i8*)
```

```
i32 f2(i8* %arg1, i32 %arg2)

BB_B:

 %4 = mul i32 %arg2, %arg2
 jne fcaller, BB_C

BB_C:

 ret call i32 @puts(%arg1)

fcaller:

 %r = call i32 f(%num, %3)
 ret %r
```

Refactoring methodology

duplicate constants

distribute globals

for every original function f

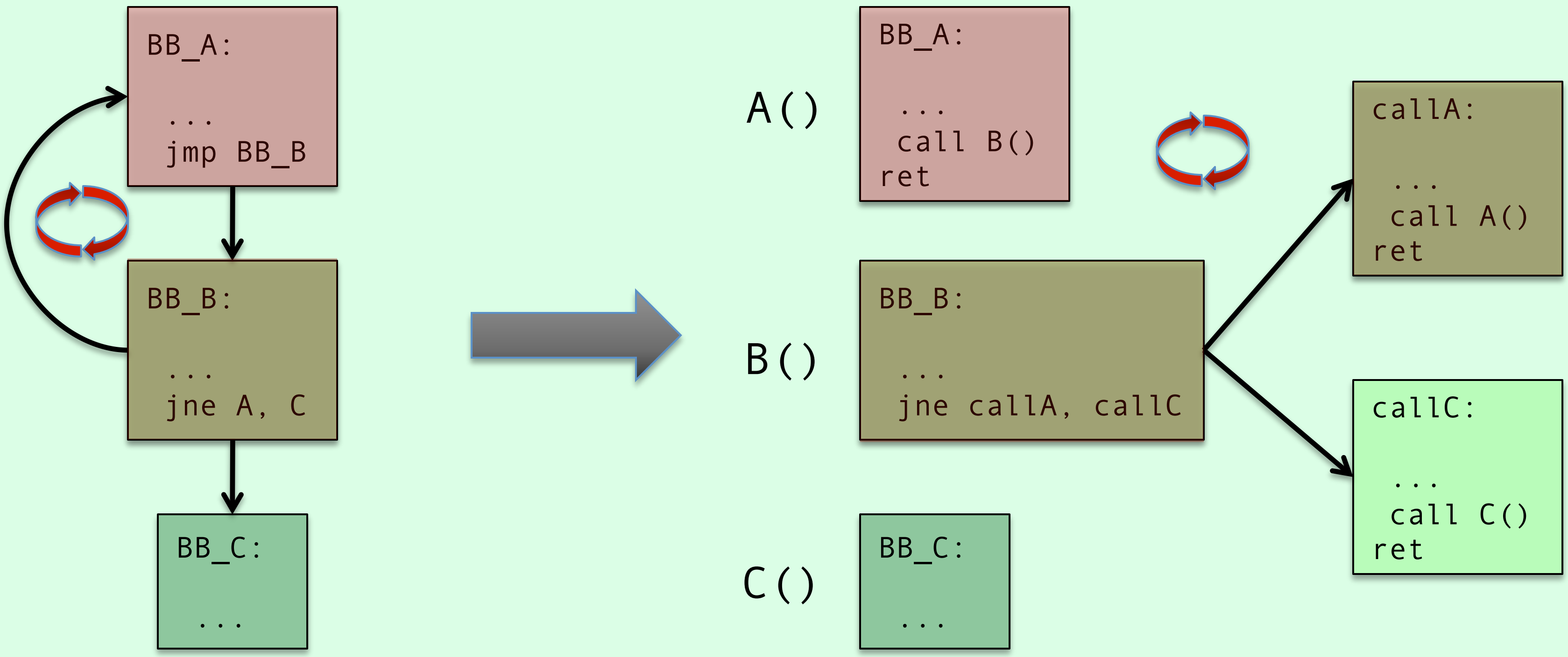
    initiatorList ← find initiators(f)

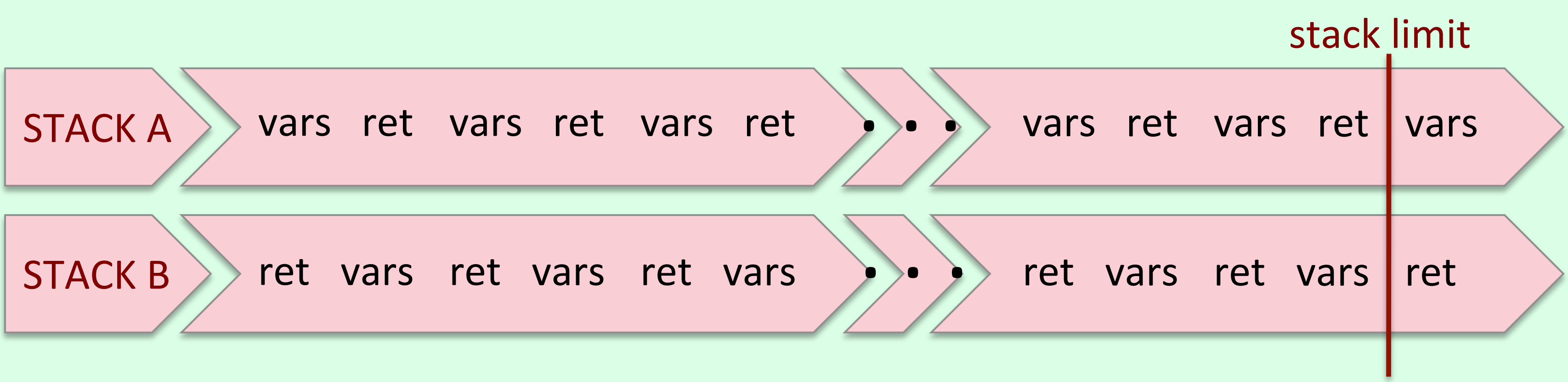
    create new functions(f, initiatorList)

    fix branches(initiatorList)

    fix phi nodes(initiatorList)

Loops generate recursive calls





Fixing loop recursion: a loop pass

```
header:

 %3 = add i32 %2, 6
 br label %latch
```

```
latch:

 %4 = mul i32 %3, %3
 %cond = icmp ne %4, 0
 br i1 %cond, label %header, label %exit
```

```
exit:

 ret call i32 @puts(%num)
```

```
preheader:

 %cmpRes = alloca i1
 store i1 true, i1* %cmpRes
 br label %header
```

```
header:

 %cond = load i1* %cmpRes
 br i1 %cond, label %postheader, label %”exit”
```

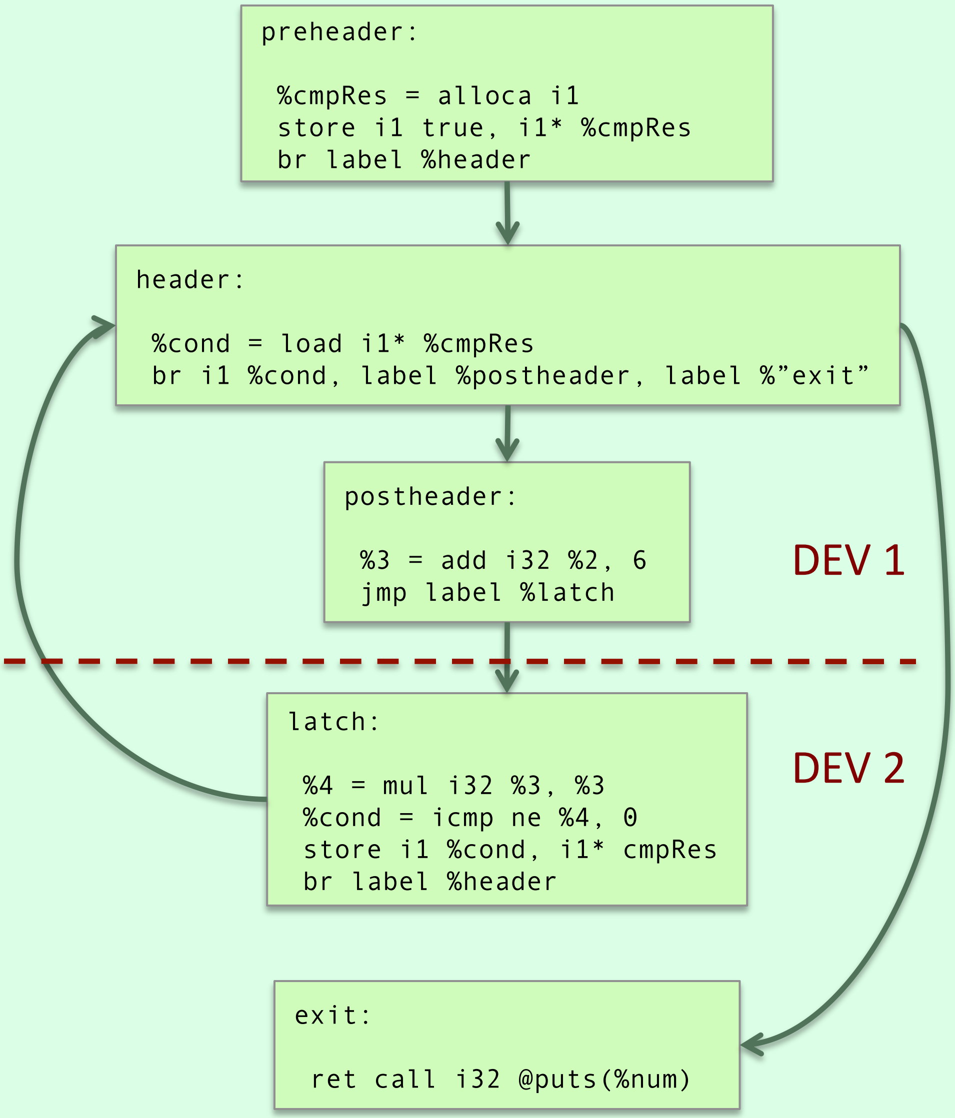
```
postheader:

 %3 = add i32 %2, 6
 jmp label %latch
```

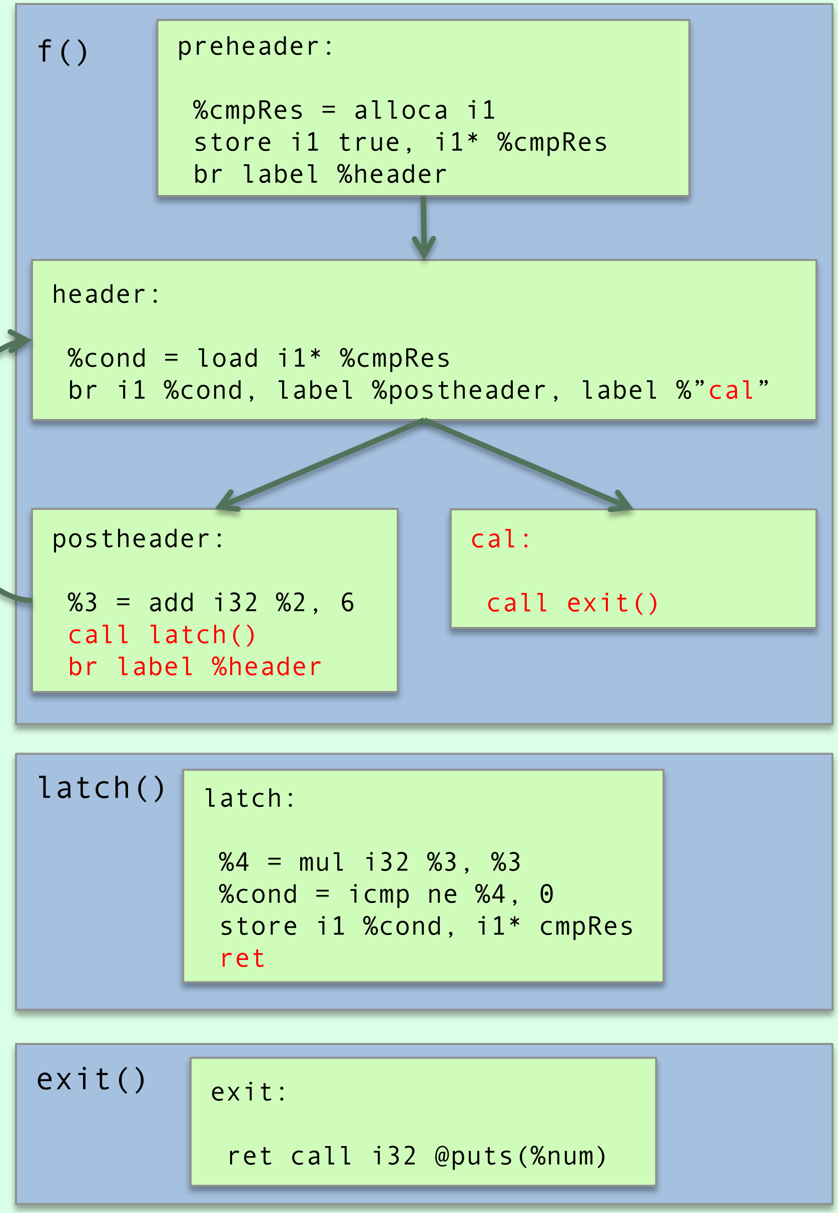
```
latch:

 %4 = mul i32 %3, %3
 %cond = icmp ne %4, 0
 store i1 %cond, i1* cmpRes
 br label %header
```

```
exit:

 ret call i32 @puts(%num)
```

Departamento de Ingeniería Electrónica
Universidad Politécnica de Madrid

Fixing loop recursion: final code refactoring



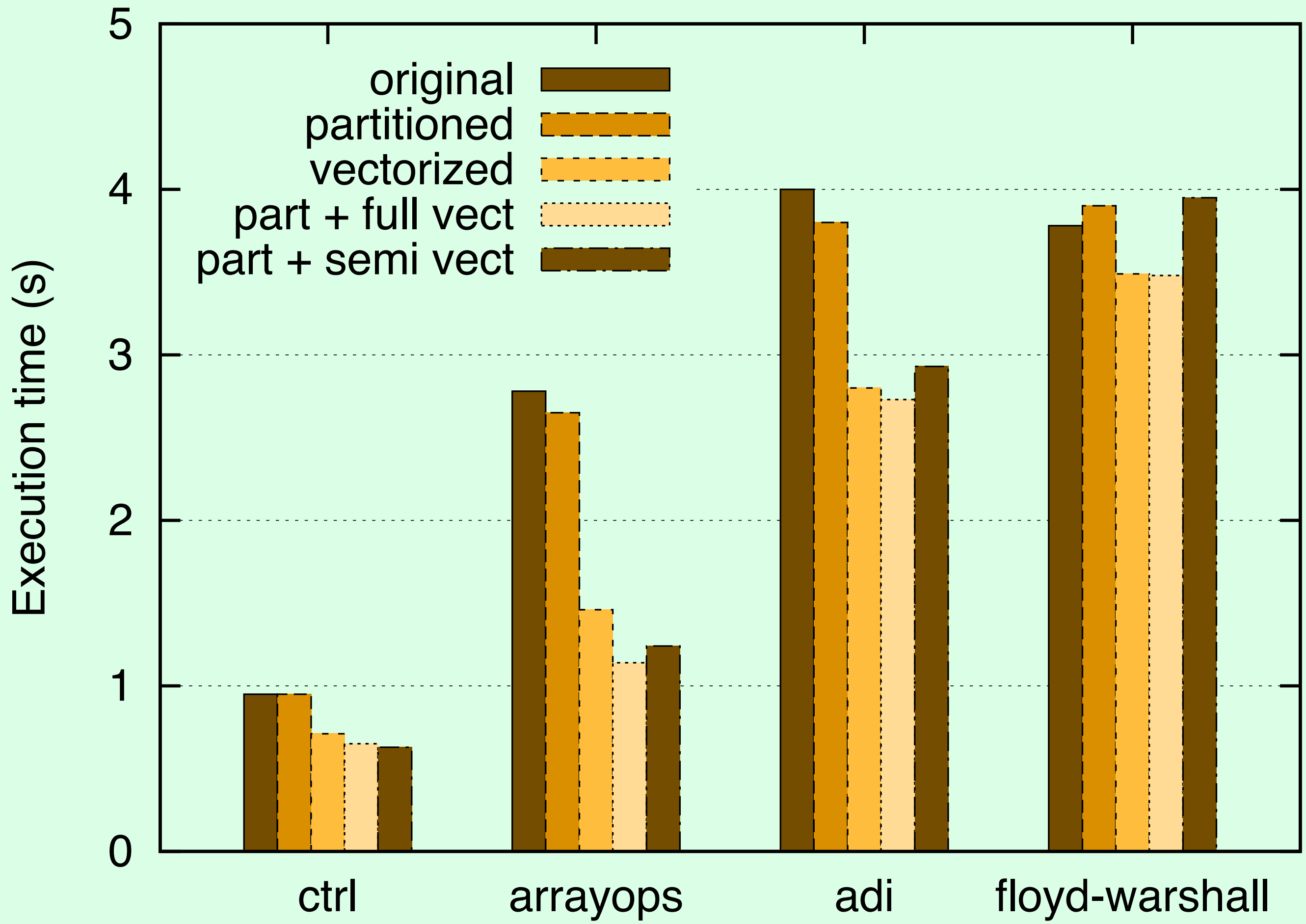


Departamento de Ingeniería Electrónica
Universidad Politécnica de Madrid

Output from the tool

```
Time profiling hello.ir
[HPCmap] Parsing module hello.ir...
[ReadArchPass] Parsing architecture ../architectures/CPU_SIMD.arch...
[EstimationPass] Estimating from profiling information...
[PartitioningPass] PARTITIONING OVERVIEW:
[PartitioningPass] Initial exec time was 1.81e-07  s, new is 1.06e-07  -- Speedup = 1.71e+00
[LoopRecursionBreakPass] Analyzing loop 5 <-> 12
[PartitionWriterPass] Generating partitioned code
PartitionWriterPass::runOnModule() -- Original module's functions:
            odd with BBs:
                        entry --> CPU
            main with BBs:
                        entry --> CPU
                        3 --> CPU
...
PartitionWriterPass::find_initiators() -- Inspecting function main()
            Trivial initiators:
                        5
                        8
            Entry block initiator: entry
            Nontrivial initiators:
                        14
...
PartitionWriterPass::create_new_Fs() -- Splitting up function main
            Function main1_CPU inserted in module CPU.part
            Moving BB 14 from function main to function main1_CPU
...
PartitionWriterPass::branches_to_fcalls() -- Fixing branches:
            to BB entry, moved to function main
            to BB 14, moved to function main1_CPU
PartitionWriterPass::fix_initiator_phis() -- Initiators:
            main2_CPU::5
                        2 phis updated
[PartitionWriterPass] Module CPU.part generated
[PartitionWriterPass] Module CPU_SIMD.part generated
Partitioned hello.ir
```

Preliminary results



Conclusions

- Compilation toolchain for heterogeneous architectures

- Code refactoring based on splitting functions into smaller ones.

- Removed recursion generated by loops being transformed into functions.

- The function call approach does not introduce a significant overhead so far.

Work in progress…

IN THE REFACTORING PASS

- Execute in a real architecture (one executable per device)
- Distributed memory
- Automatic communications

IN THE COMPLETE TOOLCHAIN

- Identification of parallelism
- Data partitioning
- Improve estimation, partitioning heuristics, profiling…

```
Time profiling hello.ir
[HPCmap] Parsing module hello.ir...
[ReadArchPass] Parsing architecture ../architectures/CPU_SIMD.arch...
[EstimationPass] Estimating from profiling information...
[PartitioningPass] Partitioning...
[PartitioningPass] PARTITIONING OVERVIEW:
[PartitioningPass] Initial exec time was 1.81e-07  s, new is 1.06e-07  -- Speedup = 1.71e+00
[LoopRecursionBreakPass] Analyzing loop 9 <-> 9
[LoopRecursionBreakPass] DONE
[LoopRecursionBreakPass] Analyzing loop 6 <-> 6
[LoopRecursionBreakPass] DONE
[PartitionWriterPass] Genera
PartitionWriterPass::runOnModule() -- Original module's functions:
            odd with BBs:
                        entry --> CPU
            main with BBs:
                        entry --> CPU
                        3 --> CPU
                        beforeHeader --> CPU
                        8 --> CPU_SIMD
                        9 --> CPU_SIMD
                        13 --> CPU
                        afterHeader --> CPU
            puts with BBs:
PartitionWriterPass::find_initiators() -- Inspecting function main()
            Trivial initiators:
                        5
                        11
            Entry block initiator: entry
            Nontrivial initiators:
                        14
            Results:
            entry has initiator entry
            beforeHeader has initiator entry
            5 has initiator 5
            11 has initiator 11
            12 has initiator 11
[PartitionWriterPass] Module CPU.part generated
[PartitionWriterPass] Module CPU_SIMD.part generated
Partitioned hello.ir
```

Departamento de Ingeniería Electrónica
Universidad Politécnica de Madrid

ETSIT UPM

Turning CFGs into callgraphs

```
Time profiling hello.ir
[HPCmap] Parsing module hello.ir...
[ReadArchPass] Parsing architecture ../architectures/CPU_SIMD.arch...
[EstimationPass] Estimating from profiling information...
[PartitioningPass] Partitioning...
[PartitioningPass] PARTITIONING OVERVIEW:
[PartitioningPass] Initial exec time was 1.81e-07  s, new is 1.06e-07  -- Speedup = 1.71e+00
[LoopRecursionBreakPass] Analyzing loop 9 <-> 9
[LoopRecursionBreakPass] DONE
[LoopRecursionBreakPass] Analyzing loop 6 <-> 6
[LoopRecursionBreakPass] DONE
[PartitionWriterPass] Genera[illegible]
PartitionWriterPass::runOnModule() -- Original module's functions:
            odd with BBs:
                        entry --> CPU
            main with BBs:
                        entry --> CPU
                        3 --> CPU
                        beforeHeader --> CPU
                        8 --> CPU_SIMD
                        9 --> CPU_SIMD
                        13 --> CPU
                        afterHeader --> CPU
            puts with BBs:
PartitionWriterPass::find_initiators() -- Inspecting function main()
            Trivial initiators:
                        5
                        11
            Entry block initiator: entry
            Nontrivial initiators:
                        14
            Results:
            entry has initiator entry
            beforeHeader has initiator entry
            5 has initiator 5
            11 has initiator 11
            12 has initiator 11
[PartitionWriterPass] Module CPU.part generated
[PartitionWriterPass] Module CPU_SIMD.part generated
Partitioned hello.ir
```

Turning CFGs into callgraphs

```
Time profiling hello.ir
[HPCmap] Parsing module hello.ir...
[ReadArchPass] Parsing architecture ../architectures/CPU_SIMD.arch...
[EstimationPass] Estimating from profiling information...
[PartitioningPass] Partitioning...
[PartitioningPass] PARTITIONING OVERVIEW:
[PartitioningPass] Initial exec time was 1.81e-07  s, new is 1.06e-07  -- Speedup = 1.71e+00
[LoopRecursionBreakPass] Analyzing loop 9 <-> 9
[LoopRecursionBreakPass] DONE
[LoopRecursionBreakPass] Analyzing loop 6 <-> 6
[LoopRecursionBreakPass] DONE
[PartitionWriterPass] Genera
PartitionWriterPass::runOnModule() -- Original module's functions:
            odd with BBs:
                        entry --> CPU
            main with BBs:
                        entry --> CPU
                        3 --> CPU
                        beforeHeader --> CPU
                        8 --> CPU_SIMD
                        9 --> CPU_SIMD
                        13 --> CPU
                        afterHeader --> CPU
            puts with BBs:
PartitionWriterPass::find_initiators() -- Inspecting function main()
            Trivial initiators:
                        5
                        11
            Entry block initiator: entry
            Nontrivial initiators:
                        14
            Results:
            entry has initiator entry
            beforeHeader has initiator entry
            5 has initiator 5
            11 has initiator 11
            12 has initiator 11
[PartitionWriterPass] Module CPU.part generated
[PartitionWriterPass] Module CPU_SIMD.part generated
Partitioned hello.ir
```

Turning CFGs into callgraphs

```
Time profiling hello.ir
[HPCmap] Parsing module hello.ir...
[ReadArchPass] Parsing architecture ../architectures/CPU_SIMD.arch...
[EstimationPass] Estimating from profiling information...
[PartitioningPass] Partitioning...
[PartitioningPass] PARTITIONING OVERVIEW:
[PartitioningPass] Initial exec time was 1.81e-07  s, new is 1.06e-07  -- Speedup = 1.71e+00
[LoopRecursionBreakPass] Analyzing [illegible] 9 <-> 9
[LoopRecursionBreakPass] DONE
[LoopRecursionBreakPass] Analyz[illegible] loop 6 <-> 6
[LoopRecursionBreakPass] DONE
[PartitionWriterPass] Genera[illegible]
PartitionWriterPass::runOnModule() -- Original module's functions:
            odd with BBs:
                        entry --> CPU
            main with BBs:
                        entry --> CPU
                        3 --> CPU
                        beforeHeader --> CPU
                        8 --> CPU_SIMD
                        9 --> CPU_SIMD
                        13 --> CPU
                        afterHeader --> CPU
            puts with BBs:
PartitionWriterPass::find_initiators() -- Inspecting function main()
            Trivial initiators:
                        5
                        11
            Entry block initiator: entry
            Nontrivial initiators:
                        14
            Results:
            entry has initiator entry
            beforeHeader has initiator entry
            5 has initiator 5
            11 has initiator 11
            12 has initiator 11
[PartitionWriterPass] Module CPU.part generated
[PartitionWriterPass] Module CPU_SIMD.part generated
Partitioned hello.ir
```

Turning CFGs into callgraphs

```
Time profiling hello.ir
[HPCmap] Parsing module hello.ir...
[ReadArchPass] Parsing architecture ../architectures/CPU_SIMD.arch...
[EstimationPass] Estimating from profiling information...
[PartitioningPass] Partitioning...
[PartitioningPass] PARTITIONING OVERVIEW:
[PartitioningPass] Initial exec time was 1.81e-07  s, new is 1.06e-07  -- Speedup = 1.71e+00
[LoopRecursionBreakPass] Analyzing loop 9 <-> 9
[LoopRecursionBreakPass] DONE
[LoopRecursionBreakPass] Analyzing loop 6 <-> 6
[LoopRecursionBreakPass] DONE
[PartitionWriterPass] Genera
PartitionWriterPass::runOnModule() -- Original module's functions:
            odd with BBs:
                        entry --> CPU
            main with BBs:
                        entry --> CPU
                        3 --> CPU
                        beforeHeader --> CPU
                        8 --> CPU_SIMD
                        9 --> CPU_SIMD
                        13 --> CPU
                        afterHeader --> CPU
            puts with BBs:
PartitionWriterPass::find_initiators() -- Inspecting function main()
            Trivial initiators:
                        5
                        11
            Entry block initiator: entry
            Nontrivial initiators:
                        14
            Results:
            entry has initiator entry
            beforeHeader has initiator entry
            5 has initiator 5
            11 has initiator 11
            12 has initiator 11
[PartitionWriterPass] Module CPU.part generated
[PartitionWriterPass] Module CPU_SIMD.part generated
Partitioned hello.ir
```

Turning CFGs into callgraphs