# OpenMP Compilation for Heterogeneous Parallel Devices

*Ph.D. Alessandro Capotondi*
*alessandro.capotondi@unimore.it*

# Heterogenous Manycores

- Ever-increasing demand for computational power has recently led to radical evolution of computer architectures

- Two design paradigms have proven effective in increasing performance and energy efficiency of compute systems
  - **Many-cores**
  - **Architectural Heterogeneity**

- A **common template** is one where a powerful general-purpose processor (**the host**) is coupled to one or more a **many-core accelerators.**

# Heterogenous Manycores

**Gyoukou**

Xeon D-1571
16C 1.3Ghz
Infiniband EDR
**PEZY-SC2**

THE GREEN 500

TOP 500 The List.

**Titan Cray X47**

Opteron 6274
16C 2.2GHz
Cray Gemini
**NVIDIA K20x**

**Tianhe-2**

Xeon E5-2692
12C 2.2GHz
TH Express-2
**Intel Xeon Phi**

**HPC / SERVER**

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

True in every computing domain and at every scale!

# Heterogenous Manycores

HPC / SERVER

SoC

**Gyoukou**

Xeon D-1571
16C 1.3Ghz
Infiniband EDR
**PEZY-SC2**

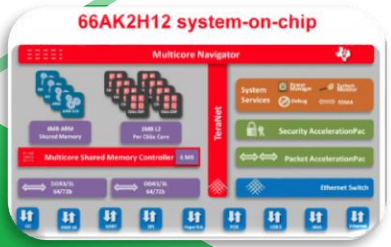
**Titan Cray X47**


Opteron 6274
16C 2.2GHz
Cray Gemini
**NVIDIA K20x**

**Tianhe-2**

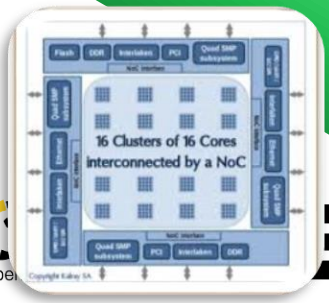
Xeon E5-2692
12C 2.2GHz
TH Express-2
**Intel Xeon Phi**

**TI Keystonell**

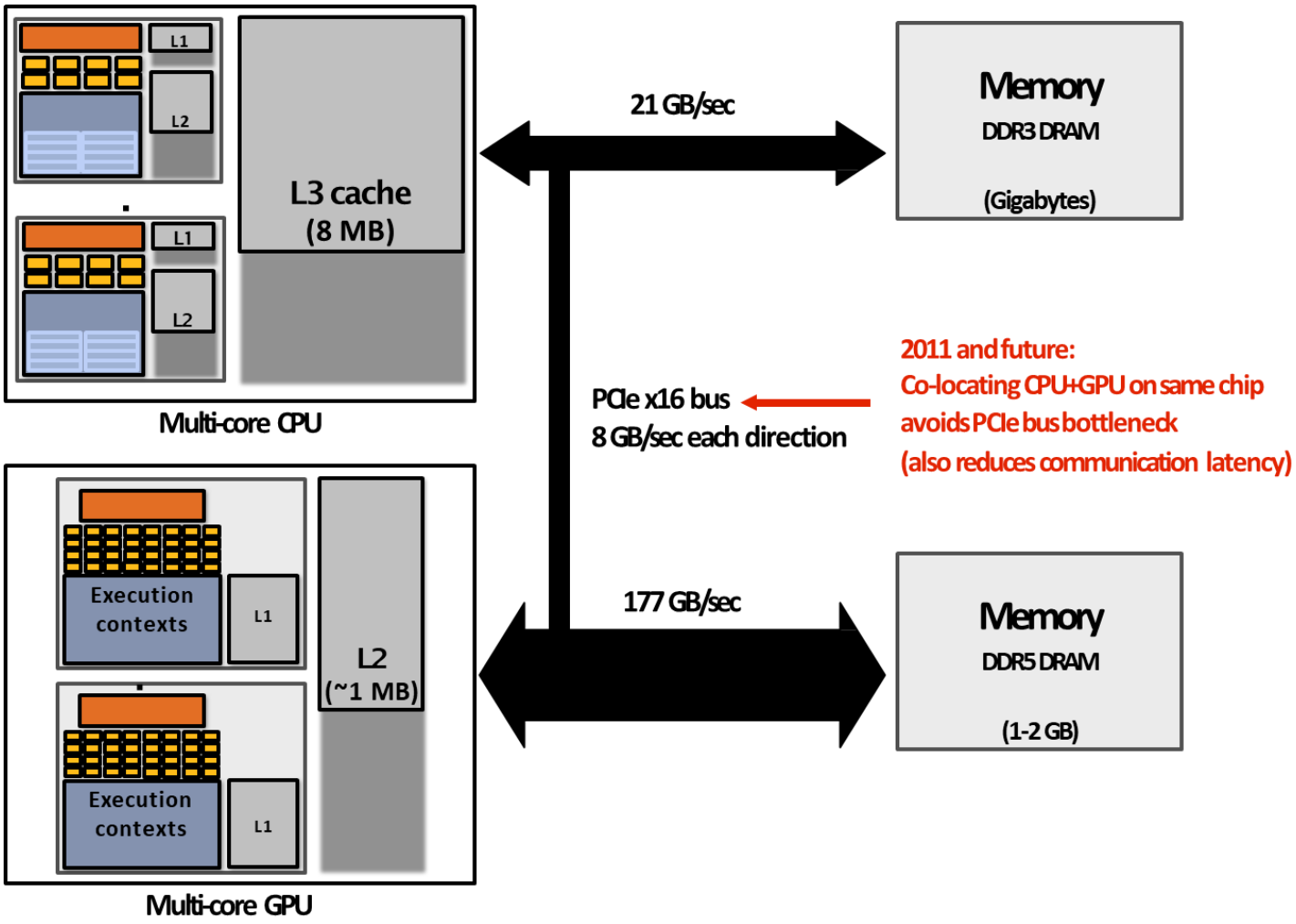



**NVIDIA Tegra TX2**

**Kalray MPPA256**



# From Heterogenous System-on-Board




PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# To Heterogeneous System-on-Board (HeSoC)

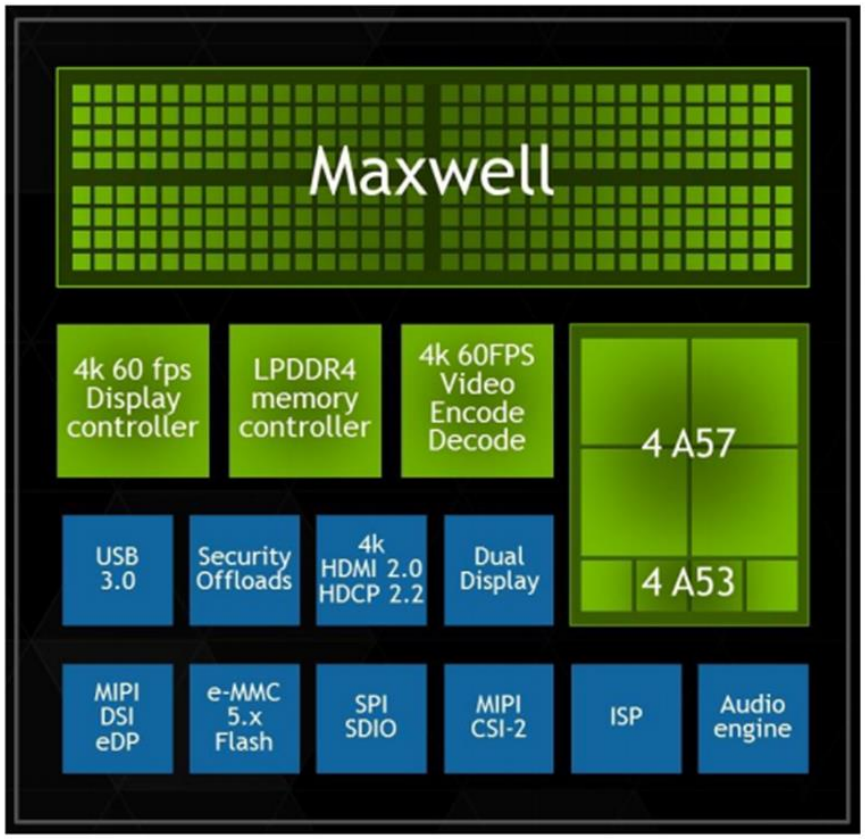
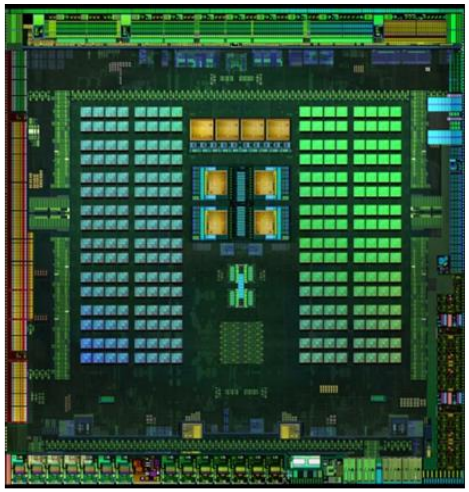



Figure 1 NVIDIA Tegra X1 Mobile Processor



## TEGRA X1 CPU CONFIGURATION

4 HIGH PERFORMANCE A57 BIG CORES

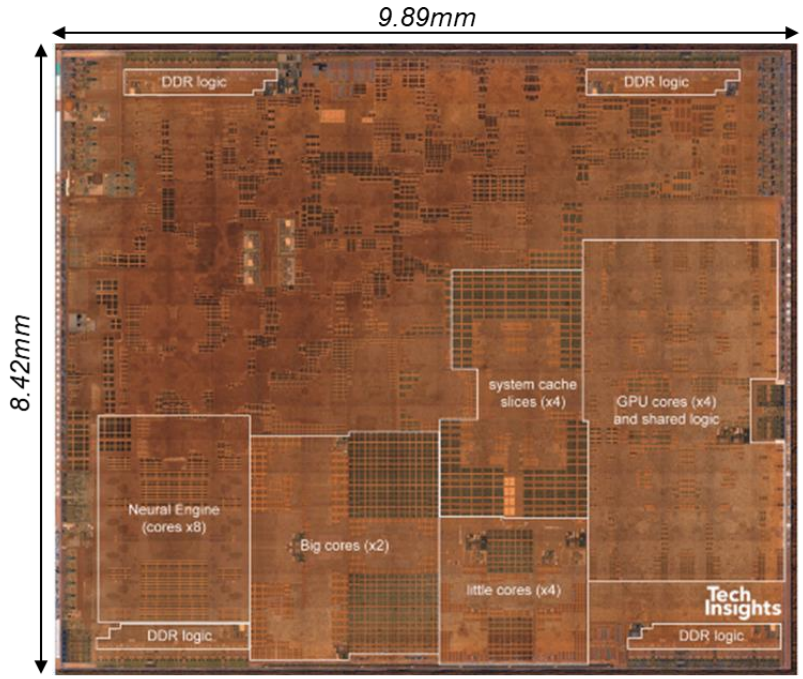
- 2MB L2 cache
- 48KB L1 instruction cache
- 32KB L1 data cache

4 HIGH EFFICIENCY A53 LITTLE CORES

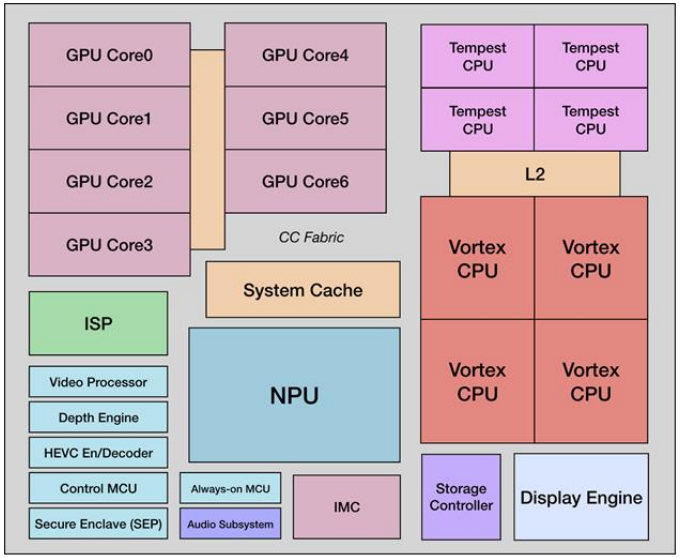
- 512KB L2 cache
- 32KB L1 instruction cache
- 32KB L1 data cache

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# …HeSOC in 2018 - Apple A12/A12X




A12 (iPhone XS) – 7nm




A12X (iPad Pro 2018)

How many processors? A lot!

*CPU:* 2/4 “big” cores + 4 “small” cores

*GPU:* 4/6 cores

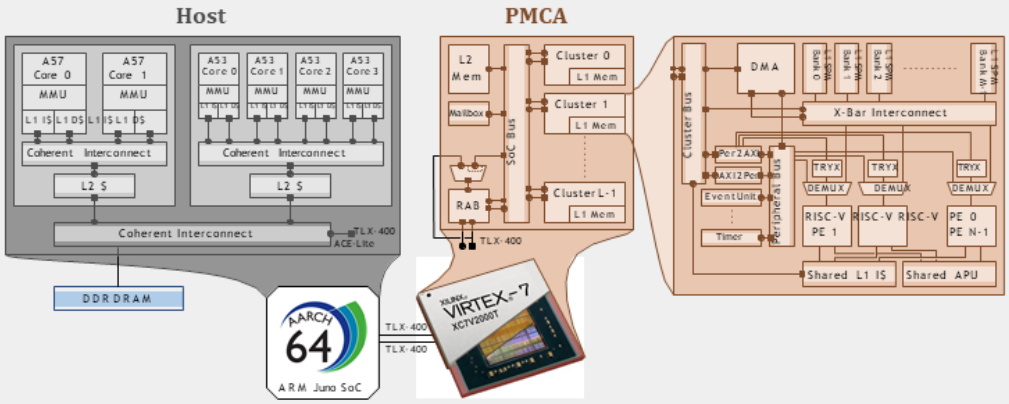
*Accelerators:* Neural Processing Unit (NPU) + Image Signal Processor (ISP)

*MCUs:* Control + Always-On

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

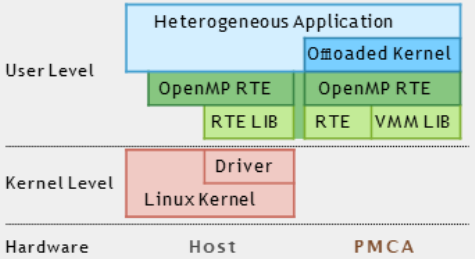
# HERO: Open-Source Heterogeneous Research Platform

## Heterogeneous Hardware Architecture




## Heterogeneous Software Stack

- single-source, single-binary cross compilation toolchain
- OpenMP 4.5
- shared virtual memory for Host and PMCA




## Profiling and automated verification solutions

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# OpenMP

- De-facto standard for shared memory programming
- Support for nested (multi-level) parallelism → *good for clusters*
- Annotations to incrementally convey parallelism to the compiler → *increased ease of use*
- Based on well-understood programming practices (shared memory, C language) → *increases productivity*

| | MM | | LU | | CONV | | PI | | Histogram | |
|---|---|---|---|---|---|---|---|---|---|---|
| | T/E | LOC | T/E | LOC | T/E | LOC | T/E | LOC | T/E | LOC |
| **OpenMP** | +++ | 45 | ++ | 60 | ++ | 70 | +++ | 32 | + | 28 |
| **TBB** | ++ | 45 | - - - | 59 | + | 53 | + | 42 | - | 58 |
| **OpenCL** | ++ | 120/108 | - - | 120/225 | - | 120/186 | - | 120/128 | - - - | 120/150 |

Table 1: Time/effort (T/E) and number of lines of code for given bench- marks for the three frameworks. MM(Matrix Multiplication), LU(LU Factorization), CONV(Image convolution). [+++ : Least time/effort - - - : Most time/effort]

*"OpenCL for programming shared memory multicore CPUs"* by Akhtar Ali , Usman Dastgeer , Christoph Kessler

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# OpenMP

- De-facto standard for shared memory programming
- Support for nested (multi-level) parallelism → *good for clusters*
- Annotations to incrementally convey parallelism to the compiler → *increased ease of use*
- Based on well-understood programming practices (shared memory, C language) → *increases productivity*
- *Since Specification 4.5 OpenMP support Heterogenous Execution Model based on offloads!*


At the moment **GCC** supports OpenMP offloading **ONLY** to:

- **Intel Xeon Phi**
- **Nvidia PTX** (only through **OpenACC**)

# OpenMP target example

```
void vec_mult()
{
  double p[N], v1[N], v2[N];

# pragma omp target map(to: v1, v2)\
                    map(from: p)
  {
#   pragma omp parallel for
    for (int i = 0; i < N; i++)
      p[i] = v1[i] * v2[i];
  }
}
```

1. Initialize target device
2. Offload target image
3. Map **TO** the device mem
4. Trigger execution target region
5. Wait termination
6. Map **FROM** the device mem

The ***compiler*** outlines the code within the target region and generates a binary version for each accelerator (multi-ISA)

The ***runtime*** libraries are in charge to:

- ***manage*** the accelerator devices
- ***map*** the variables
- ***run/wait*** execution of target regions

# OpenMP target – Howto Manage multiple ISA? Compilation

1. During the **ipa_write_summaries** pass the intermediate representation of outlined functions is streamed out into the **.gnu.offload_lto_*** sections of the "**fat**" object file. This object file also may contain **.gnu.lto_*** sections for the regular link-time optimizations.

2. Also the **decls** from **offload_funcs** and **offload_vars** are streamed out into the **.gnu.offload_lto_.offload_table** section. Later an accel compiler will read this section to produce target's mapping table.

3. In **omp_finish_file** function the addresses from **offload_funcs** and **offload_vars** are written into the **.gnu.offload_funcs** and **.gnu.offload_vars** sections correspondingly.

# OpenMP target – Howto Manage multiple ISA? Linking

1. When all source files are compiled, pre-linker driver **collect2** is invoked. It runs the linker, which loads linker plugin **liblto_plugin.so**, which runs **lto-wrapper**. Without offloading the lto-wrapper is called for link-time recompilation if at least one object file contains **.gnu.lto_* sections**. If some files contain offloading, then linker plugin will execute lto-wrapper even if there are no **.gnu.lto_*** sections. Offloading without linker plugin is not supported.

2. **lto-wrapper** runs **mkoffload** for each accel target, specified during the configuration.

3. **mkoffload** runs accel compiler, which **reads IR** from the **.gnu.offload_lto_*** sections and compiles it for the accel target. Then **mkoffload packs this target code (image**) into the special section of a new host's object file. The object file produced with **mkoffload** should contain a constructor that calls **GOMP_offload_register{,_ver}** to identify itself at run-time. Arguments to that function are a symbol called **__OFFLOAD_TABLE__** (provided by libgcc and unique per shared object), a target identifier, and some other data needed for a particular target (a pointer to the image, a table with information about mappings between host and offload functions and variables).

4. Linker adds new object files, produced by **mkoffload**, to the list of host's input object files

## NVIDIA Tesla K20

- 13 Multiprocessors
- 2496 CUDA Cores
- 5 GB of global memory
- GPU clock rate 760MHz


## Intel MIC Xeon Phi

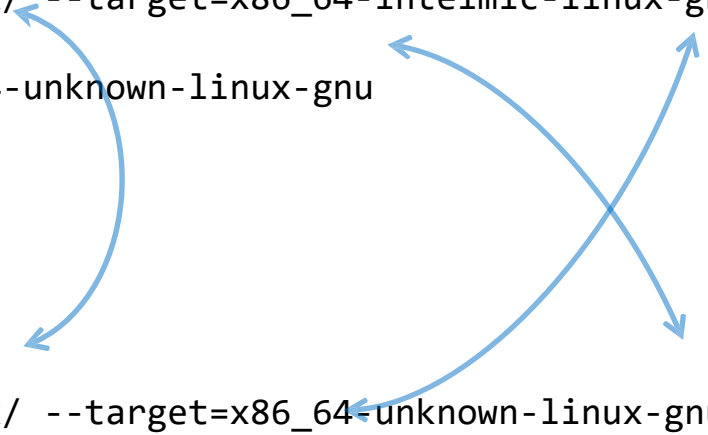
- 236 compute units
- 8 GB of global memory
- CPU clock rate 1052 MHz


- 61 cores
- 512bit-SIMD units

# *Implementation in GCC*
# *Building host and target compilers*

## Intel MIC target (accel/offload/offloading) compiler:

```
$ configure --prefix=/install/prefix/ --target=x86_64-intelmic-linux-gnu
\
  --enable-as-accelerator-for=x86_64-unknown-linux-gnu

$ make && make install
```



## Host compiler:

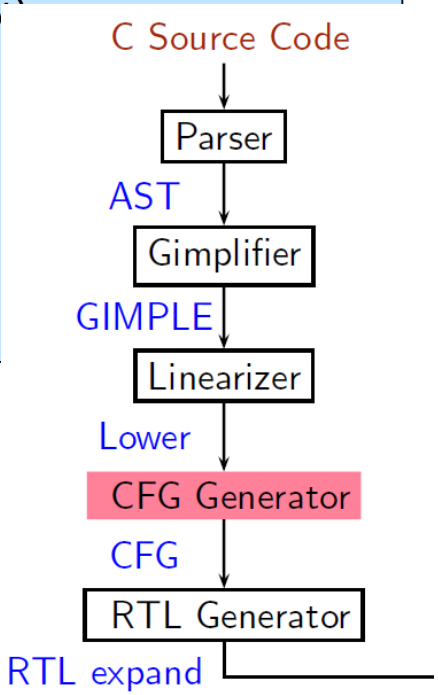
```
$ configure --prefix=/install/prefix/ --target=x86_64-unknown-linux-gnu \
  --enable-offload-targets=x86_64-intelmic-linux-gnu

$ make && make install
```

More info: https://gcc.gnu.org/wiki/Offloading

# Implementation in GCC
# test1.c

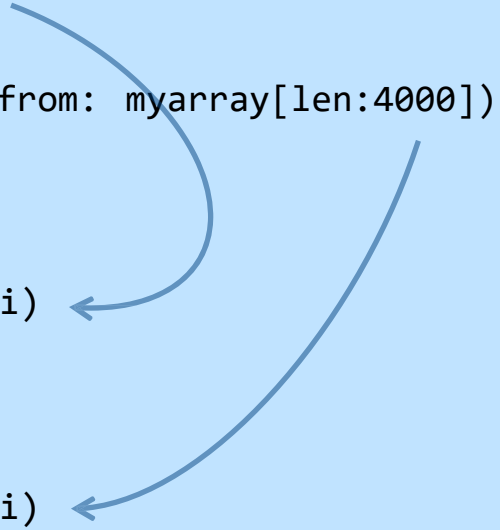
```
int foo ()
{
  int myvar = 1;
  int myarray[1000];

  #pragma omp target device(0) map(tofrom: myvar)
    { myvar *= 10; }

  #pragma omp target device(0) map(to: myvar) map(from: myarray)
    { myarray[42] = myvar; }

  return myarray[42];
}

int main ()
{
  return foo ();
}
```

```
$ gcc –fopenmp –c test1.c –o test1.o
```



# Implementation in GCC
# test1.c.004t.gimple

```
int foo ()
{
  int myvar = 1;
  int myarray[1000];

  #pragma omp target device(0) map(tofrom: myvar[len:4])
    { myvar *= 10; }

  #pragma omp target device(0) map(to: myvar[len:4]) map(from: myarray[len:4000])
    { myarray[42] = myvar; }

  return myarray[42];
}

foo._omp_fn.0 (struct .omp_data_t.3 & restrict .omp_data_i)
{
  int *D.1873 = .omp_data_i->myvar;
  *D.1873 = *D.1873 * 10;
}

foo._omp_fn.1 (struct .omp_data_t.4 & restrict .omp_data_i)
{
  int *D.1868 = .omp_data_i->myvar;
  int *D.1870 = .omp_data_i->myarray;
  *D.1870[42] = *D.1868;
}
```

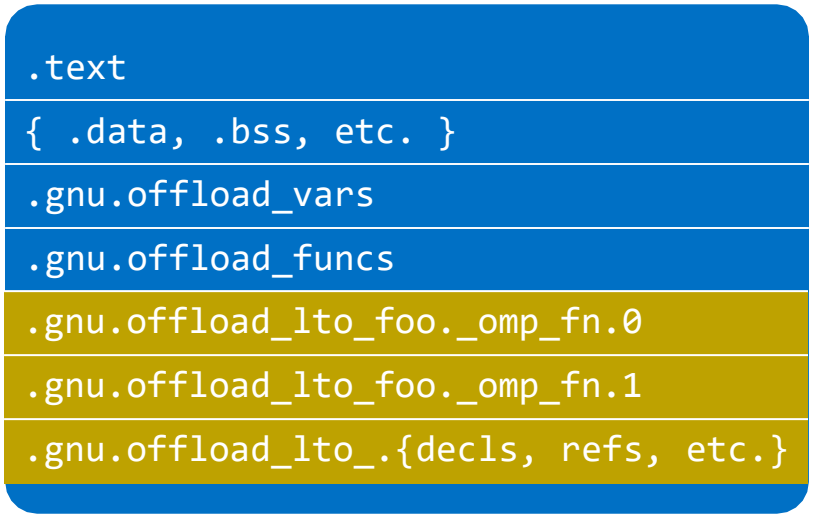


# Implementation in GCC
# test1.c.012t.ompexp

```
int foo ()
{
  .omp_data_arr.myvar = &myvar;
  .omp_data_sizes[1] = { 4 };
  .omp_data_kinds[1] = { GOMP_MAP_TOFROM };

  __builtin_GOMP_target (0, foo._omp_fn.0, 1, &.omp_data_arr,
  &.omp_data_sizes,
                         &.omp_data_kinds);
  .omp_data_arr.myvar   = &myvar;
  .omp_data_arr.myarray = &myarray;
  .omp_data_sizes[2] = { 4, 4000 };
  .omp_data_kinds[2] = { GOMP_MAP_TO, GOMP_MAP_FROM };

  __builtin_GOMP_target (0, foo._omp_fn.1, 2, &.omp_data_arr, &.omp_data_sizes,
                         &.omp_data_kinds);
}

foo._omp_fn.0 (struct .omp_data_t.3 & restrict
.omp_data_i)
{
  int *D.1873 = .omp_data_i->myvar;
  *D.1873 = *D.1873 * 10;
}

foo._omp_fn.1 (struct .omp_data_t.4 & restrict
.omp_data_i)
{
  int *D.1868 = .omp_data_i->myvar;
  int *D.1870 = .omp_data_i->myarray;
  *D.1870[42] = *D.1868;
}
```

# *Implementation in GCC*
# *test1.o*

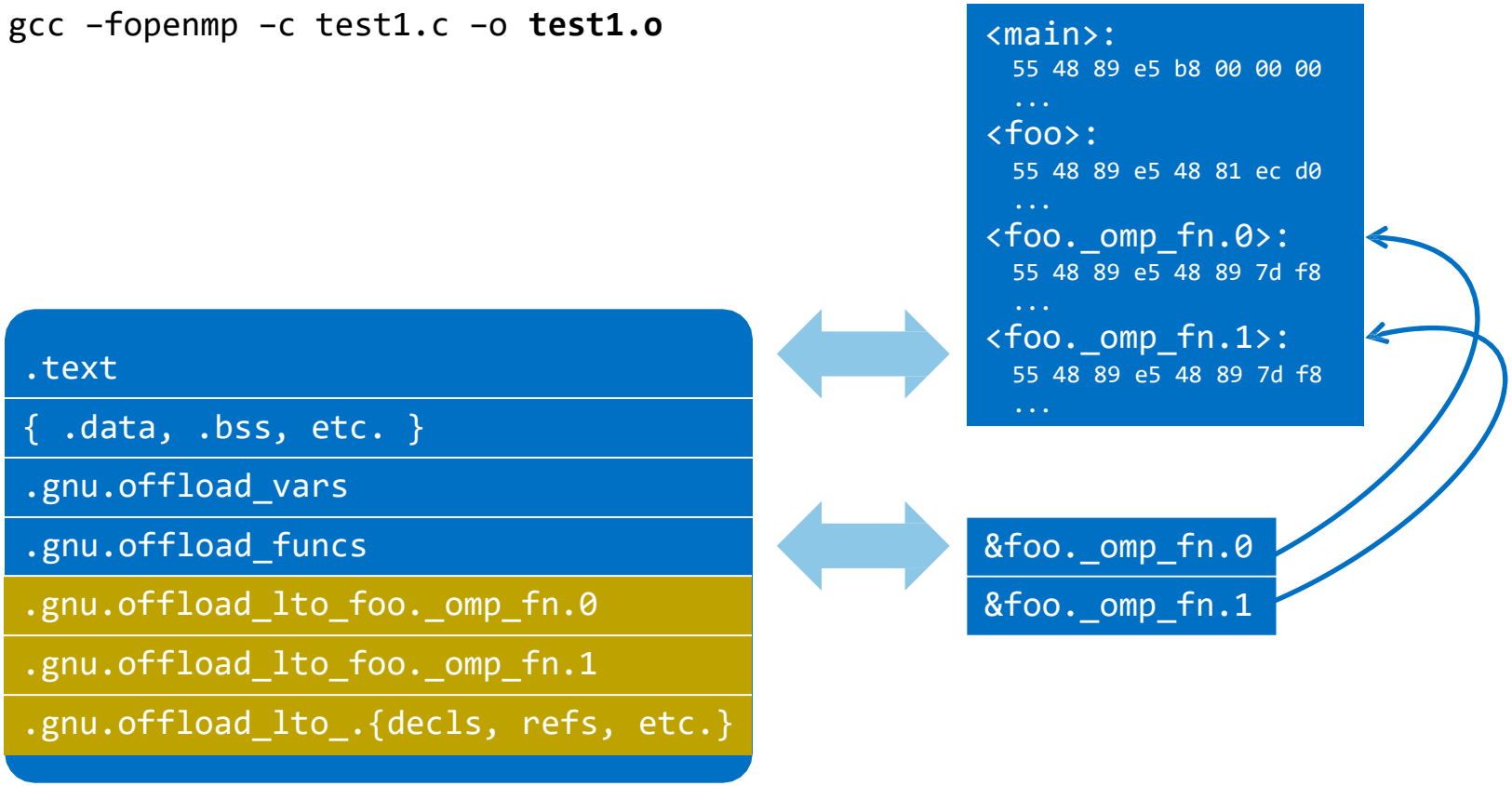
```
$ gcc –fopenmp –c test1.c –o test1.o
```




Fat object, similar to general LTO (https://gcc.gnu.org/onlinedocs/gccint/LTO-object-file-layout.html)

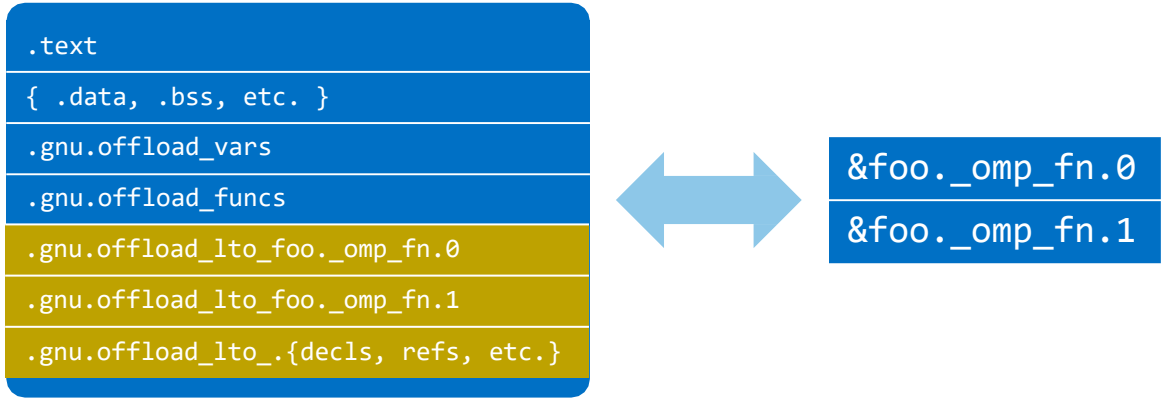
# *Implementation in GCC test1.o*

```
$ gcc –fopenmp –c test1.c –o test1.o
```

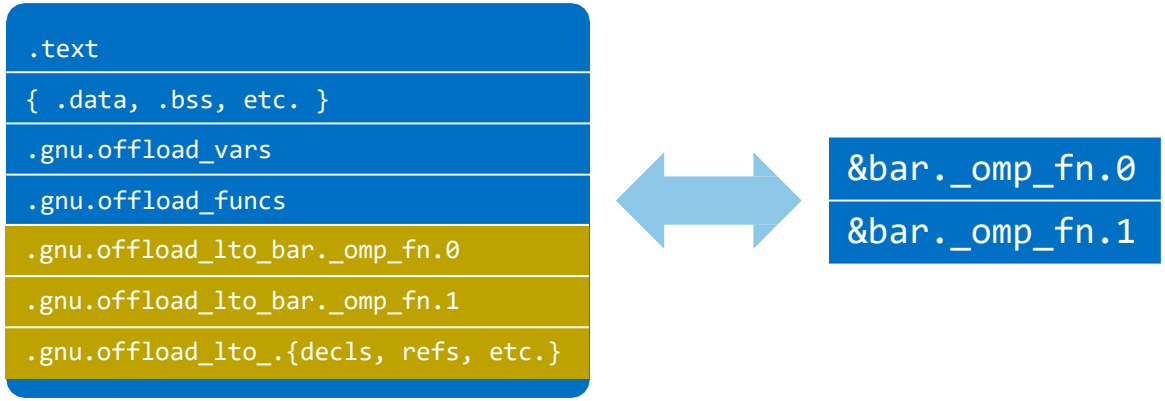


# *Implementation in GCC test1.o and test2.o*

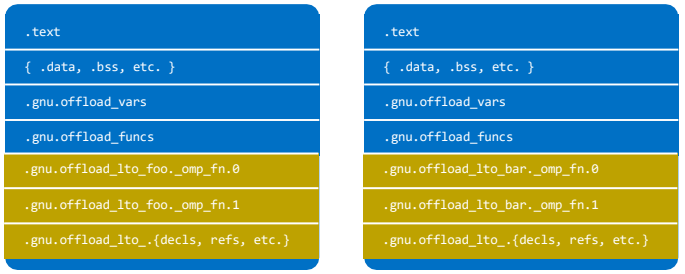
```
$ gcc –fopenmp –c test1.c –o test1.o
```




```
$ gcc –fopenmp –c test2.c –o test2.o
```

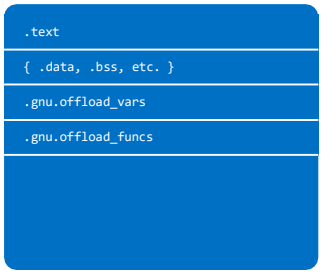


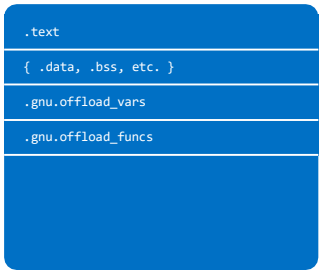
# *Implementation in GCC Link-time compilation*



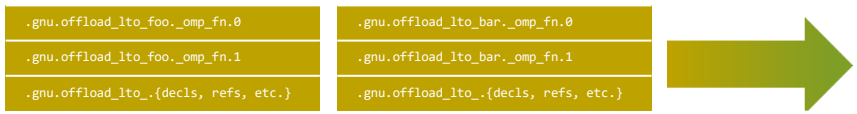

```
$ gcc –fopenmp test1.o test2.o –o a.out
    |
  collect2
    |
  collect-ld
    |
  ld
    |
  liblto_plugin.so
    |
  lto-wrapper
    |
  intelmic-mkoffload
    |
  intelmic-gcc
```

# *Implementation in GCC*
# *Link-time compilation*

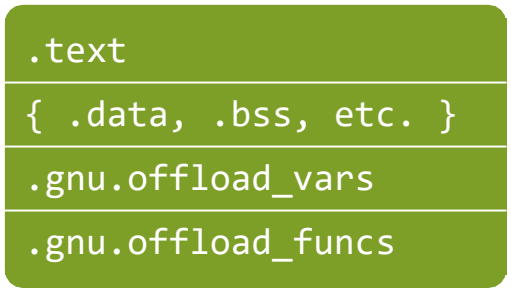


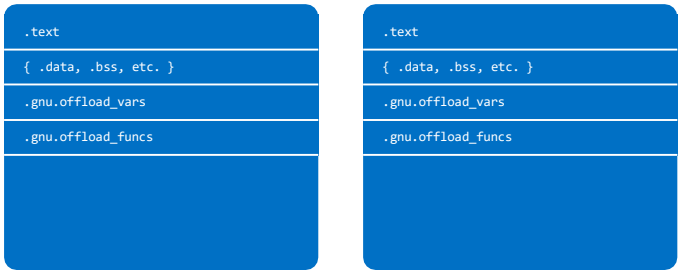

```
$ gcc –fopenmp test1.o test2.o –o a.out
    |
  collect2
    |
  collect-ld
    |
  ld
    |
  liblto_plugin.so
    |
  lto-wrapper
    |
  intelmic-mkoffload
    |
  intelmic-gcc
```



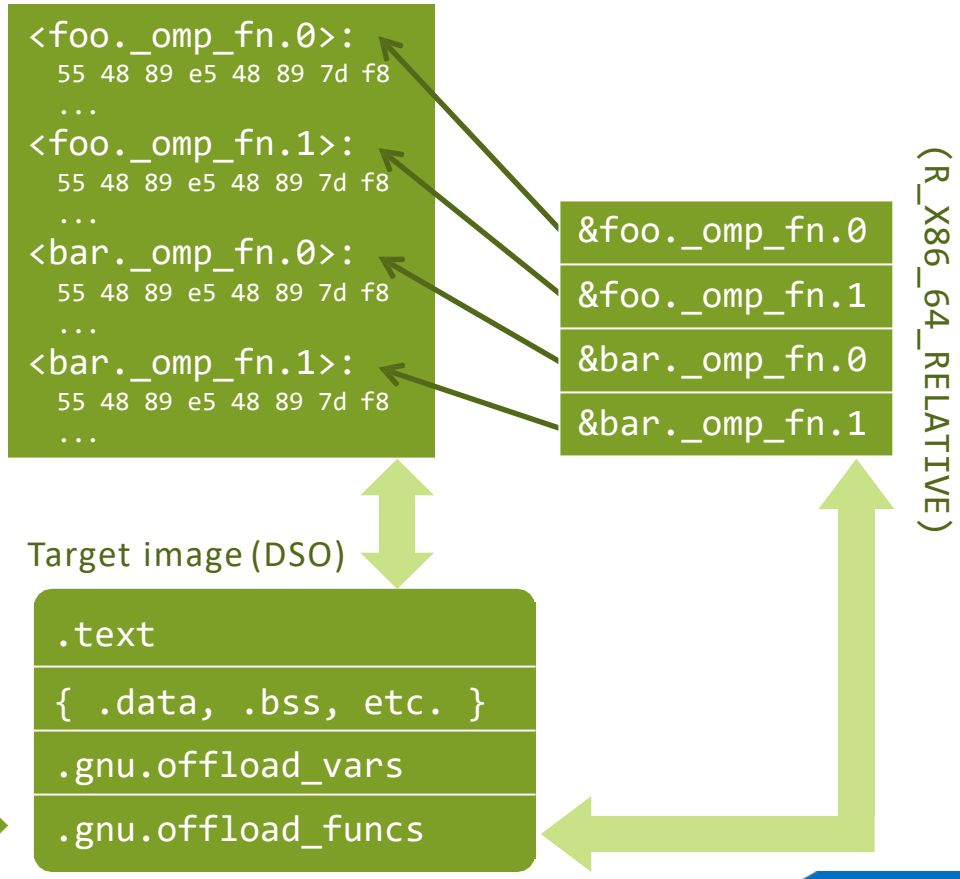

Target image (DSO)



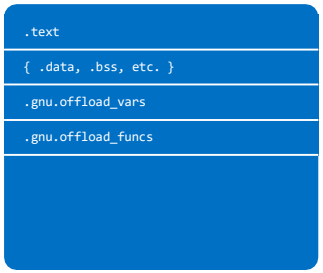
# *Implementation in GCC Link-time compilation*

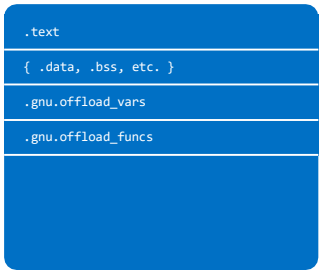



```
$ gcc –fopenmp test1.o test2.o –o a.out
    |
  collect2
    |
  collect-ld
    |
  ld
    |
  liblto_plugin.so
    |
  lto-wrapper
    |
  intelmic-mkoffload
    |
  intelmic-gcc
```

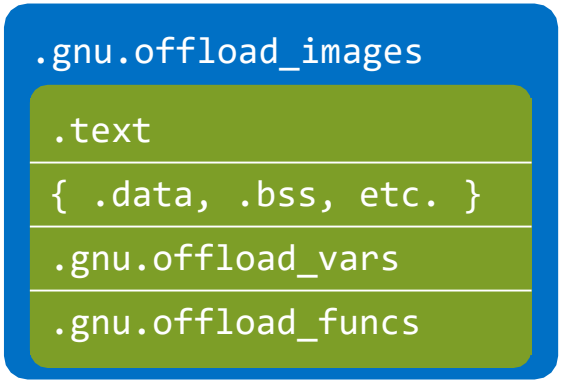


# *Implementation in GCC Link-time compilation*



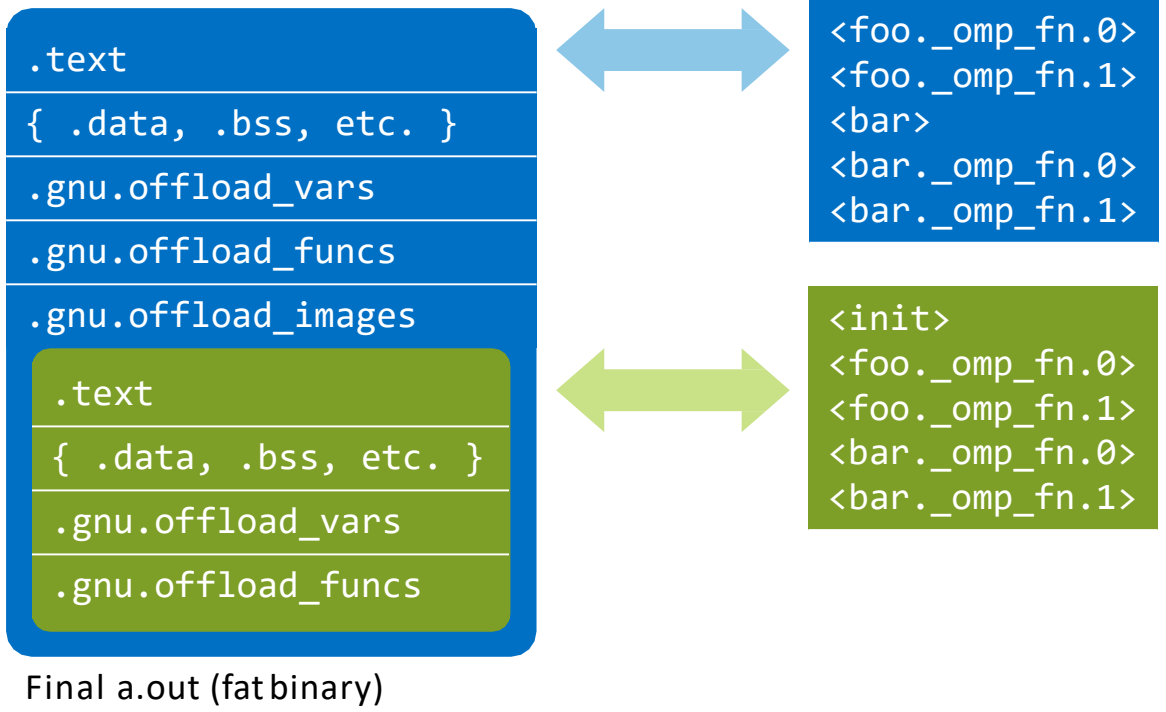




```
$ gcc –fopenmp test1.o test2.o –o a.out
    |
  collect2
    |
  collect-ld
    |
  ld
    |
  liblto_plugin.so
    |
  lto-wrapper
    |
  intelmic-mkoffload
    |
  intelmic-gcc
```



# Implementation in GCC Linking

```
$ gcc –fopenmp test1.o test2.o –o a.out
    |
  collect2
    |
  collect-ld
    |
  ld
    |
  liblto_plugin.so
    |
  lto-wrapper
    |
  intelmic-mkoffload
    |
  intelmic-gcc
```

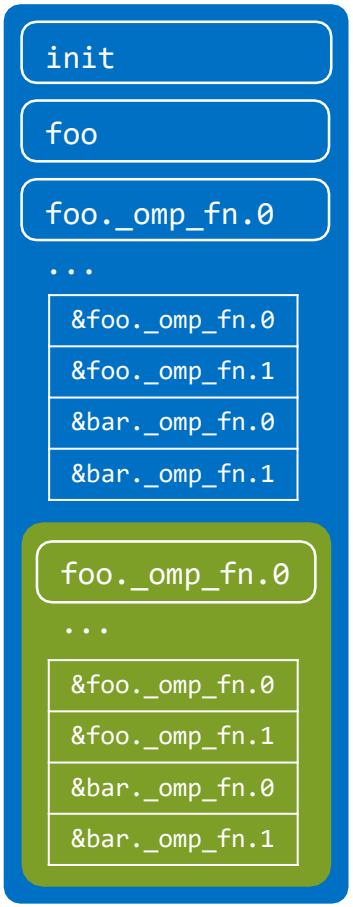



Final a.out (fat binary)

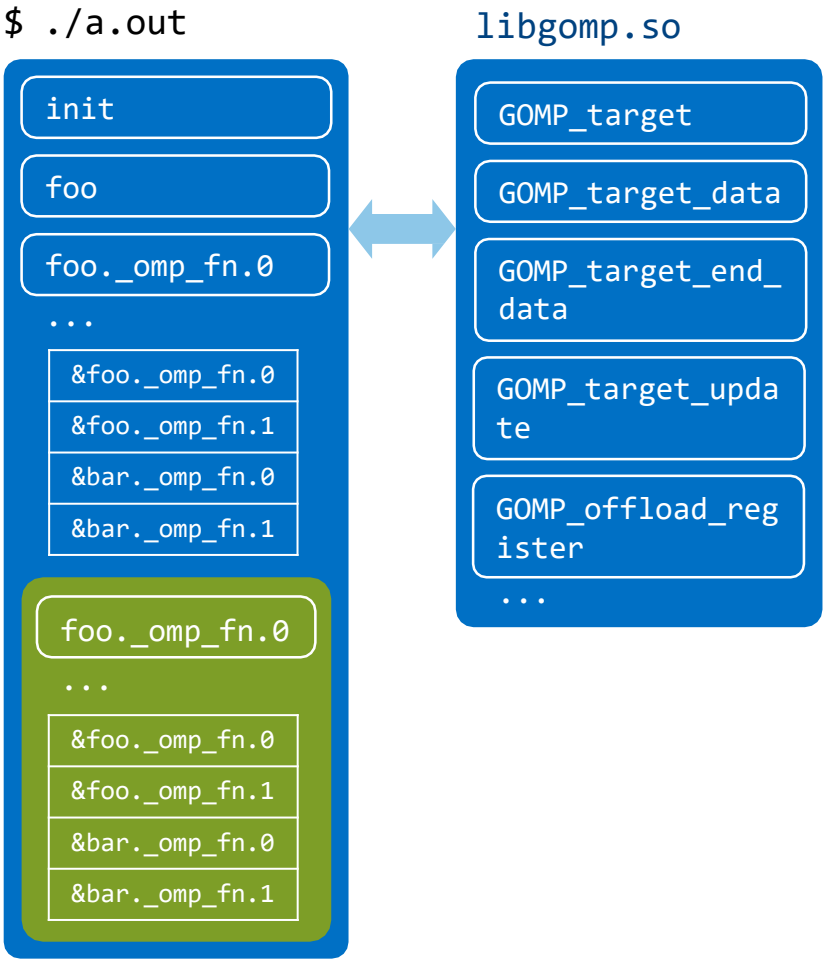
# *Implementation in GCC. Execution*

# Implementation in GCC Startup

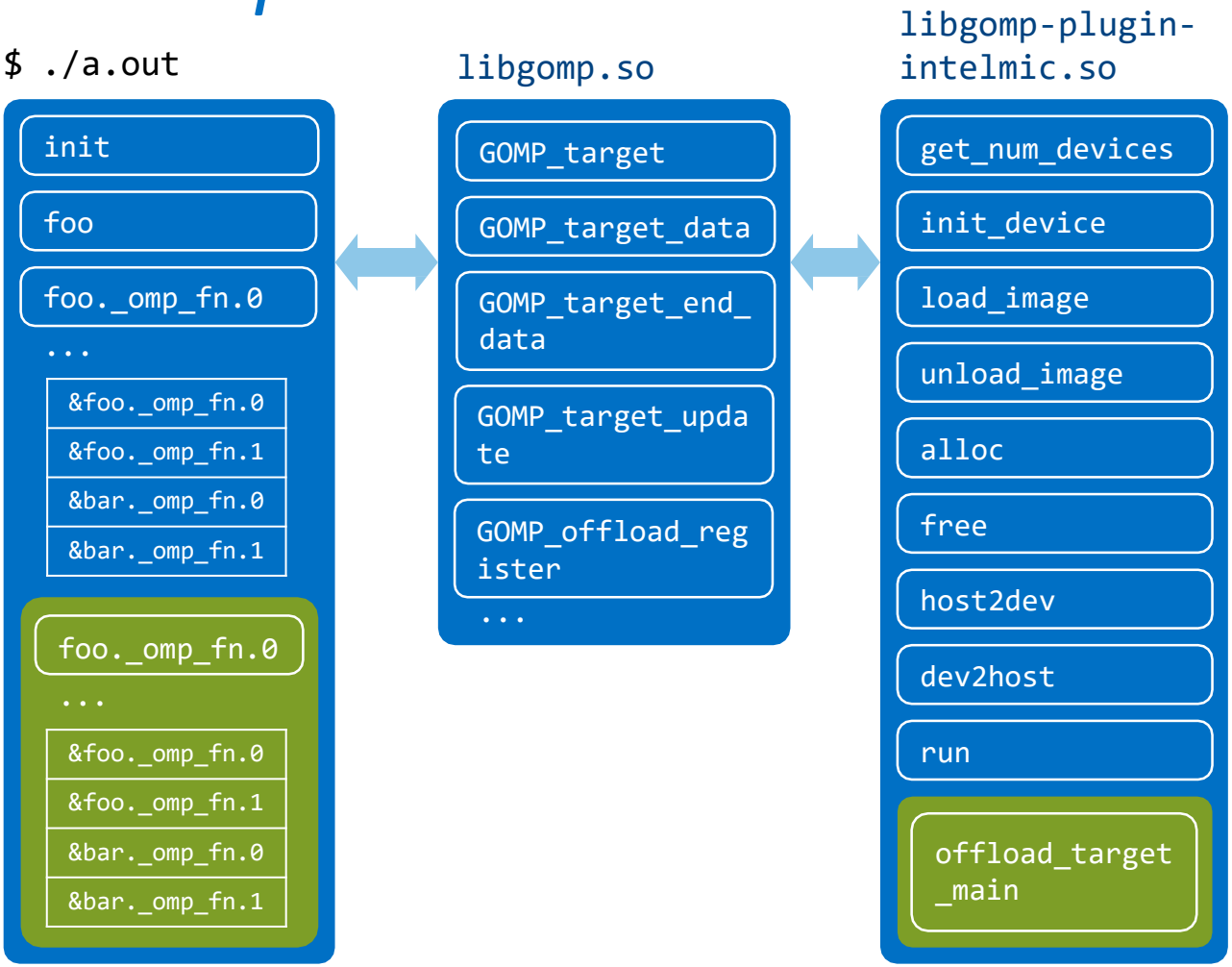
```
$ ./a.out
```



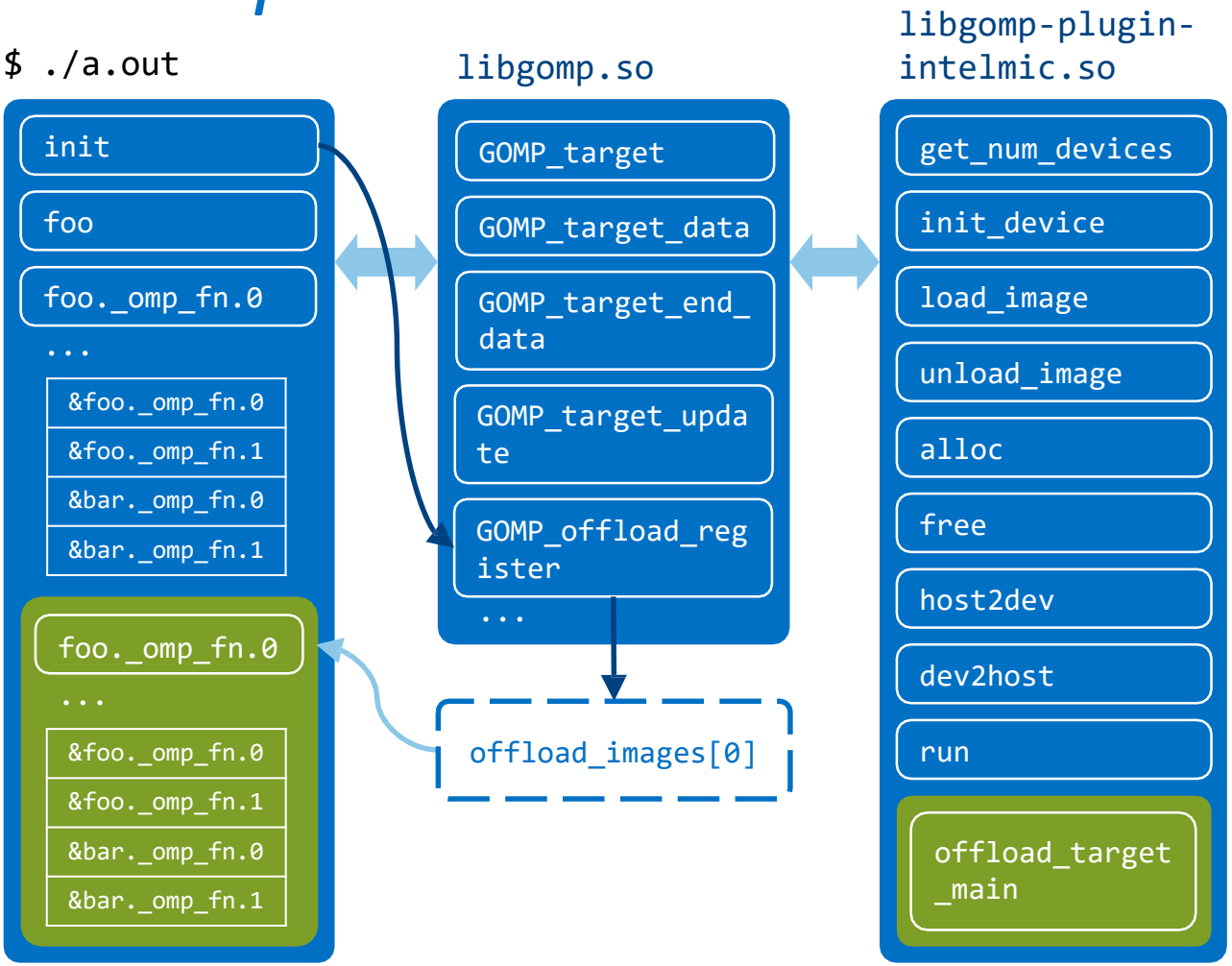
# Implementation in GCC Startup



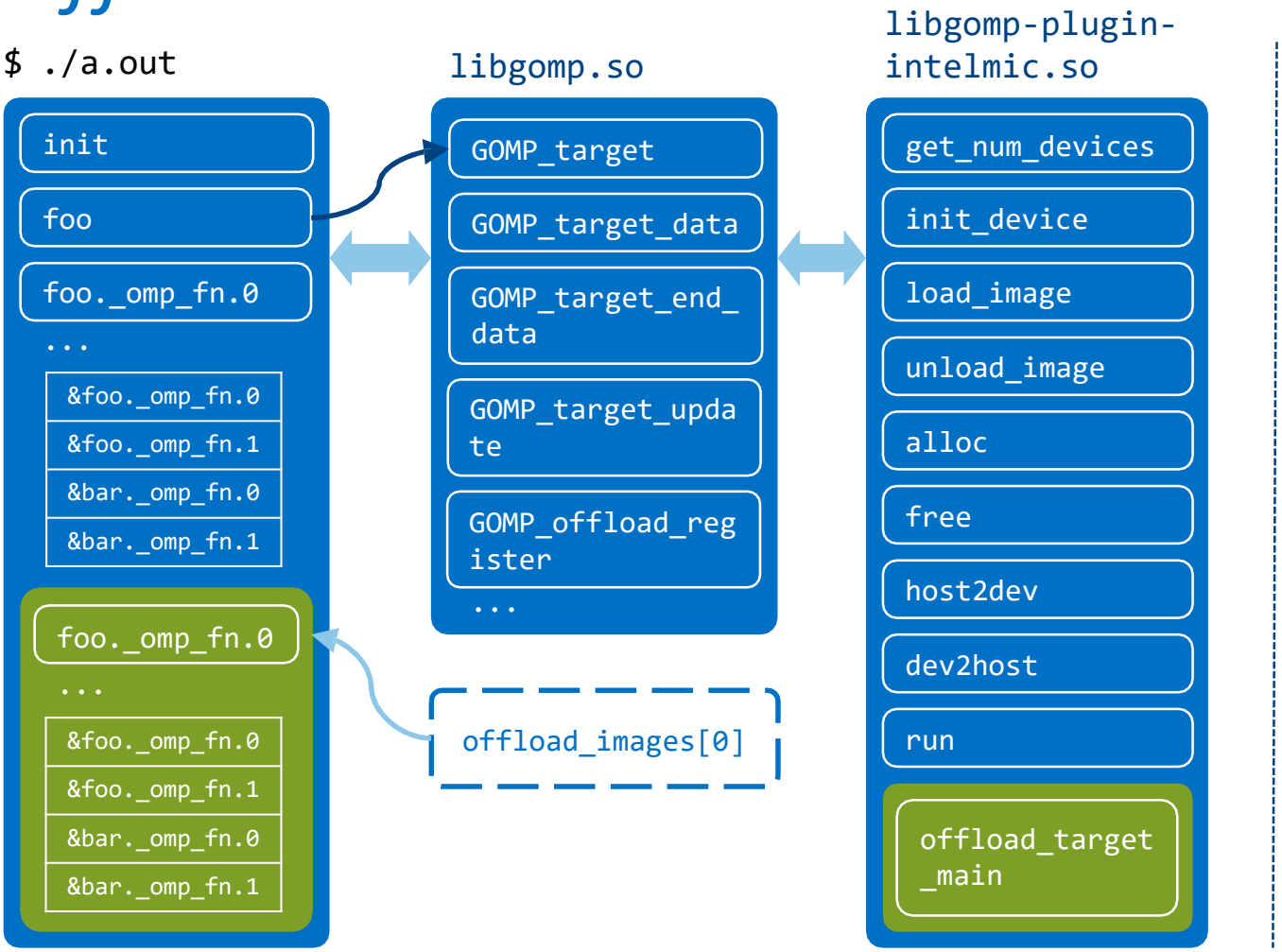
# Implementation in GCC Startup



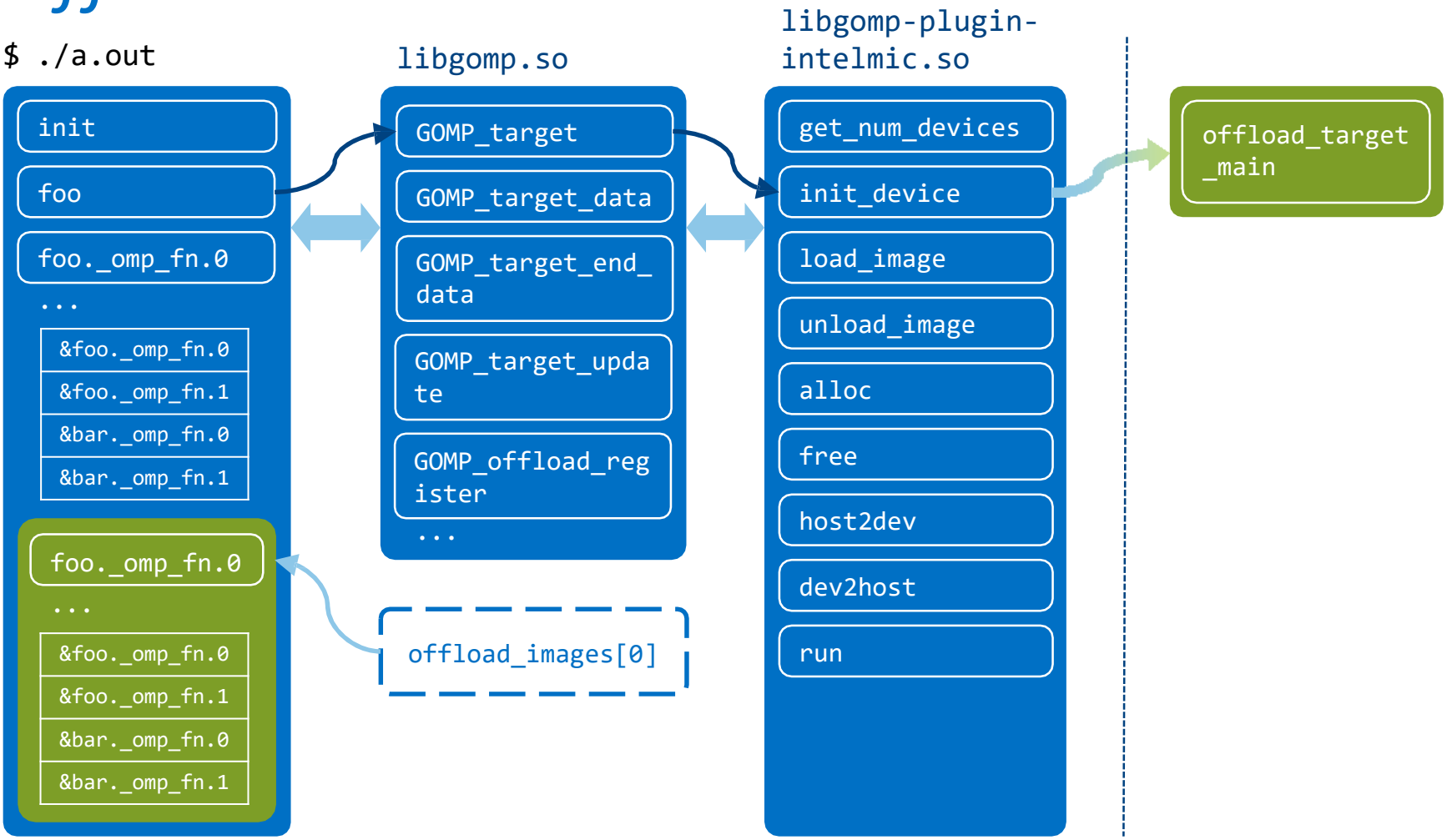
# Implementation in GCC Startup



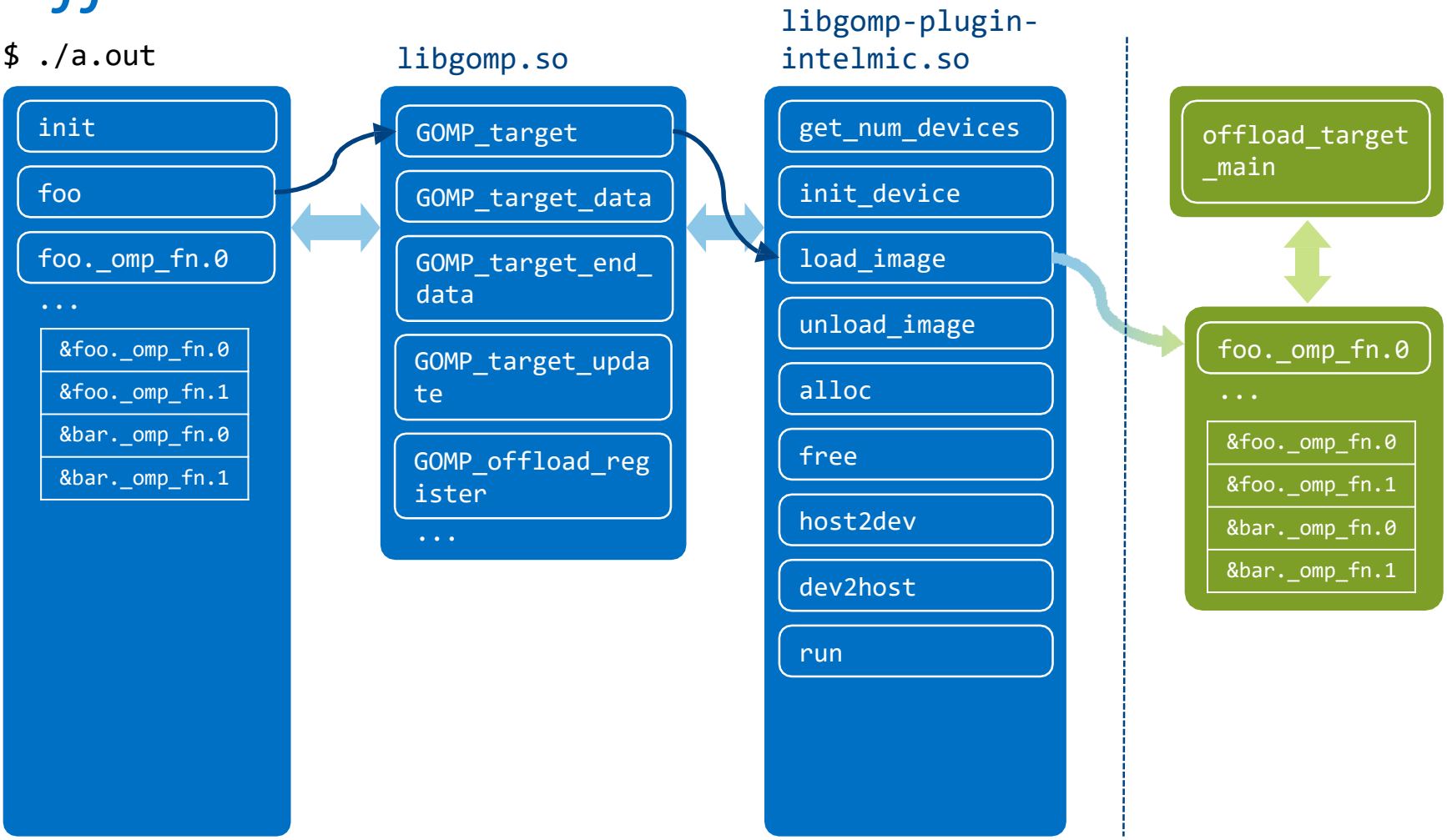
# Implementation in GCC
# Offload initialization



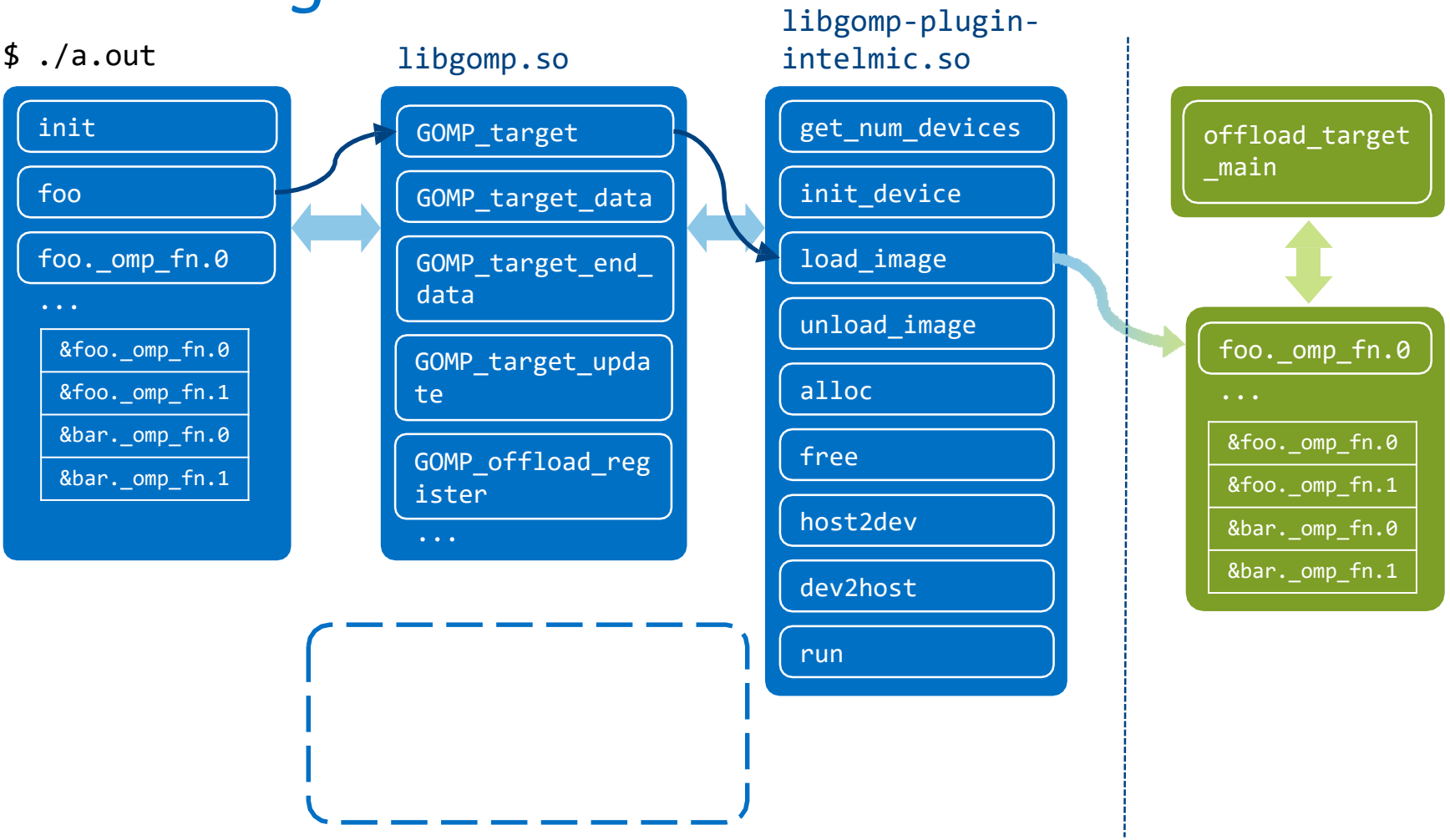
# Implementation in GCC
# Offload initialization



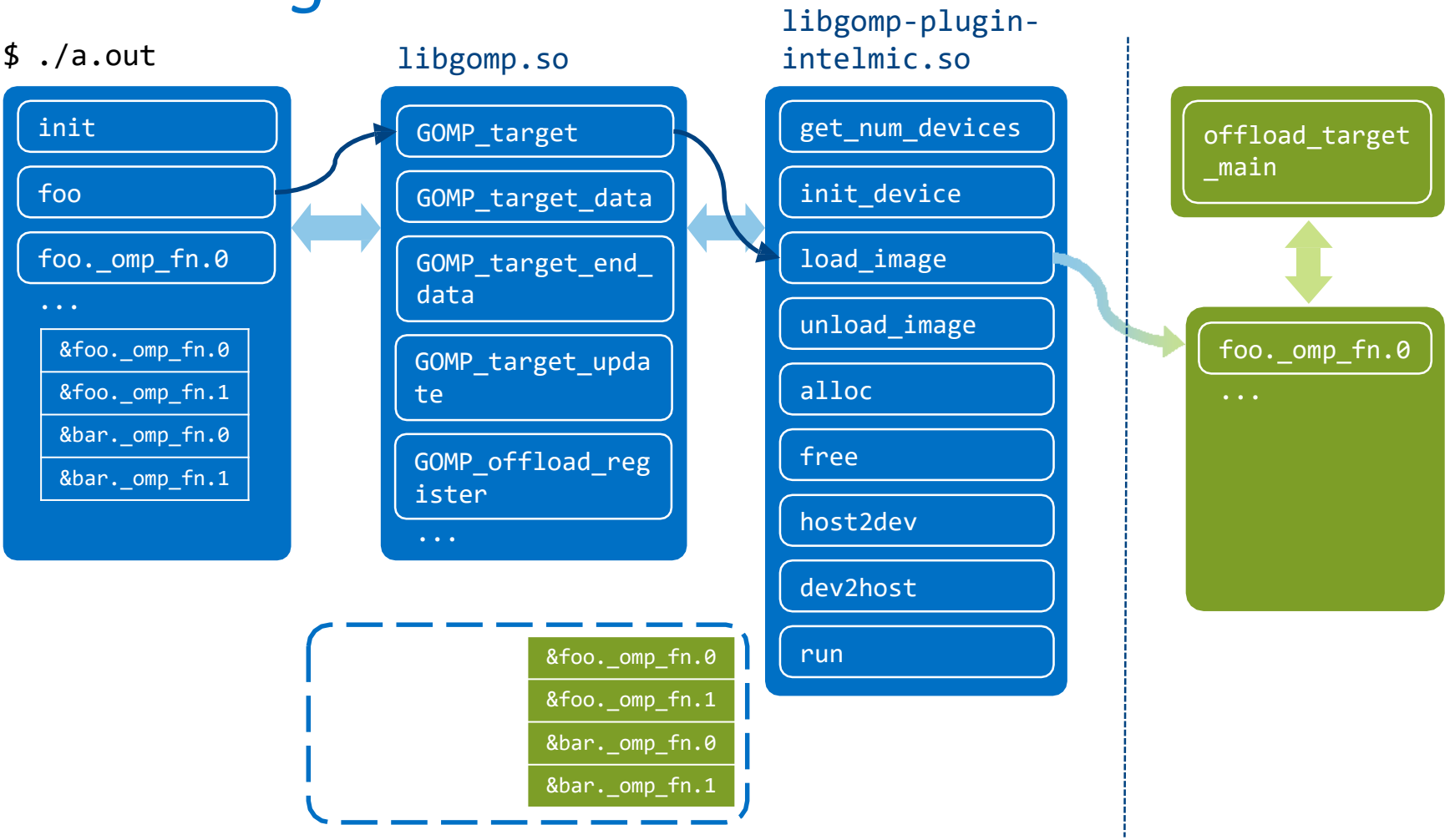
# Implementation in GCC
# Offload initialization



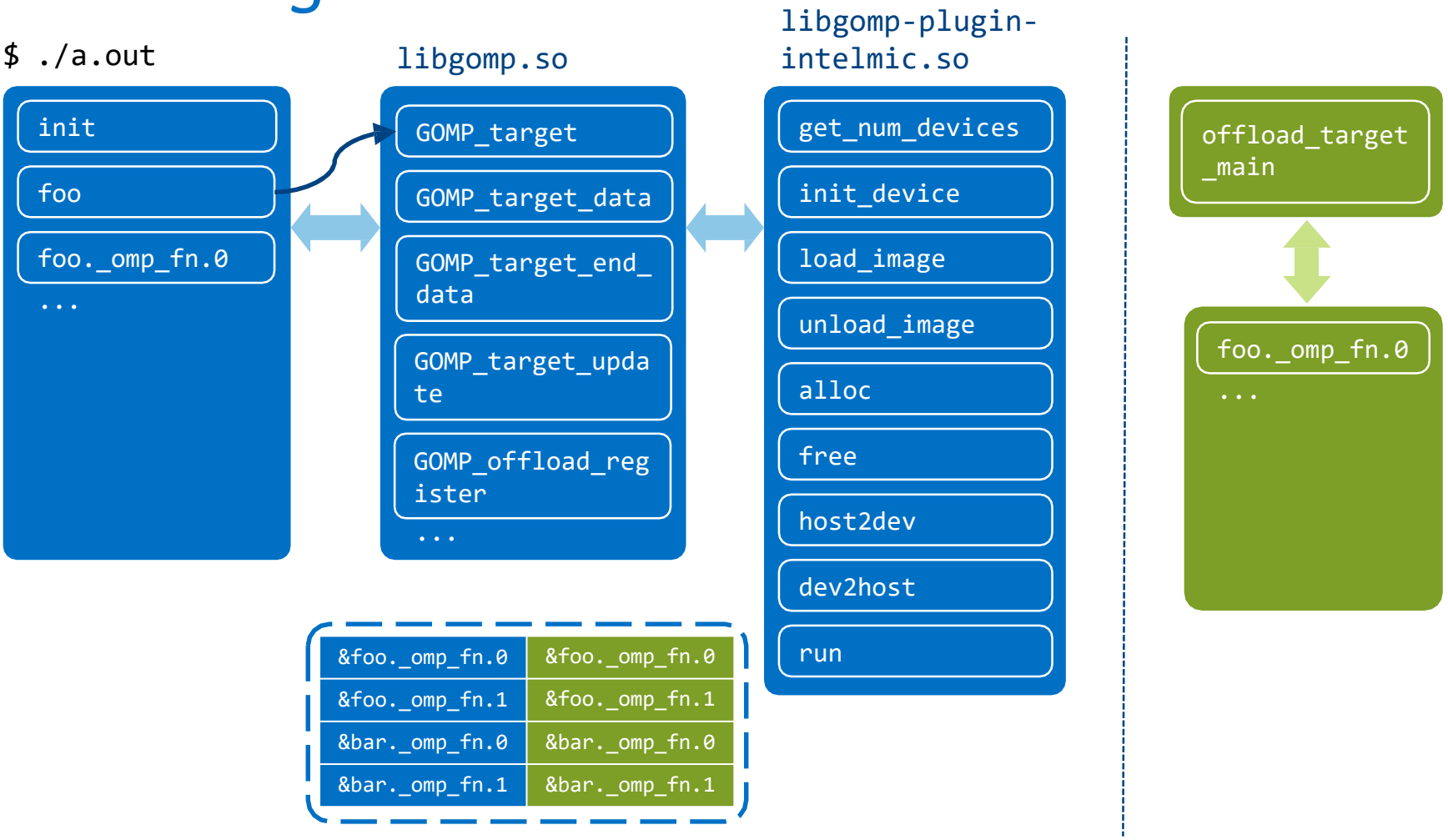
# Implementation in GCC
# Matching tables



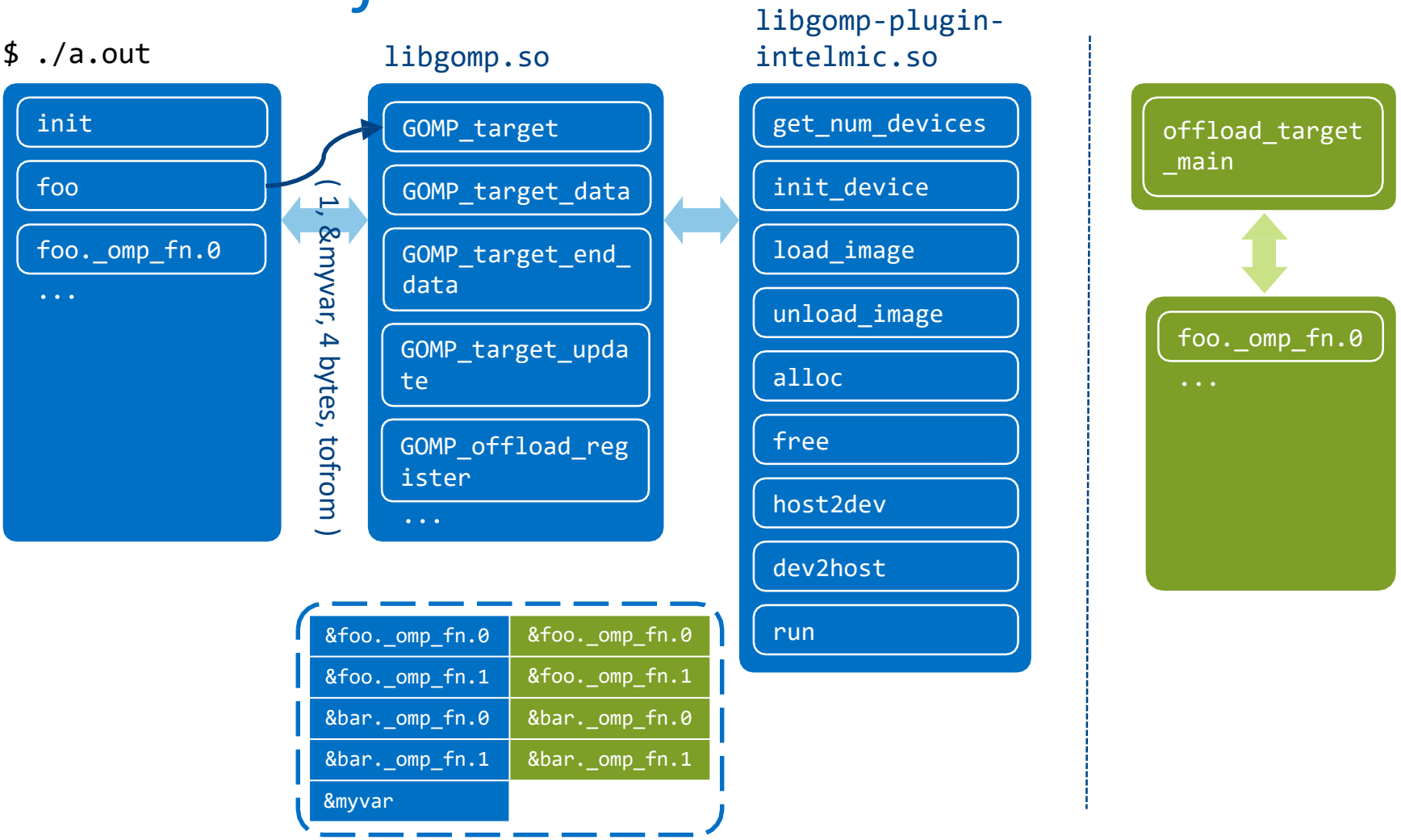
# Implementation in GCC
# Matching tables



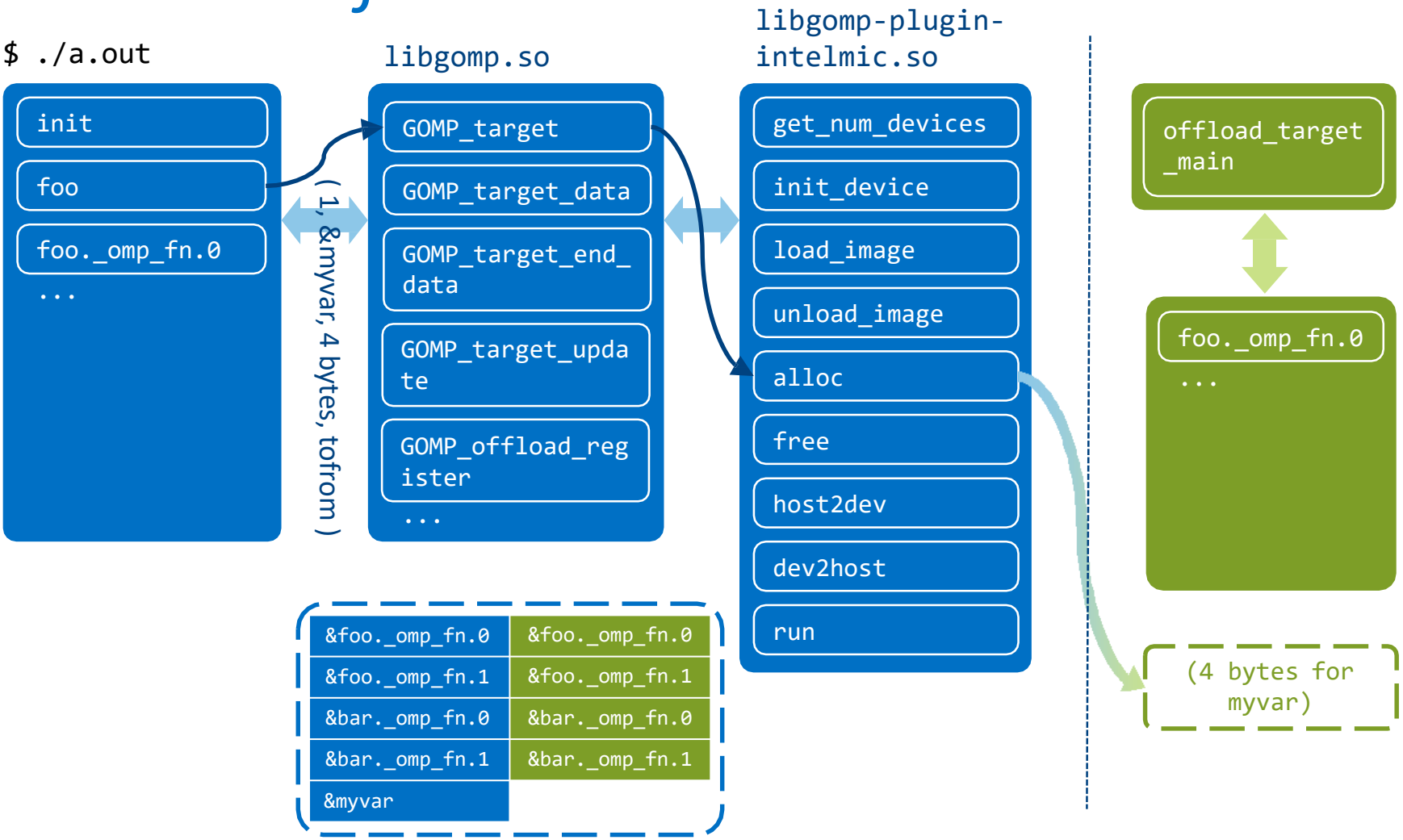
# Implementation in GCC
# Matching tables



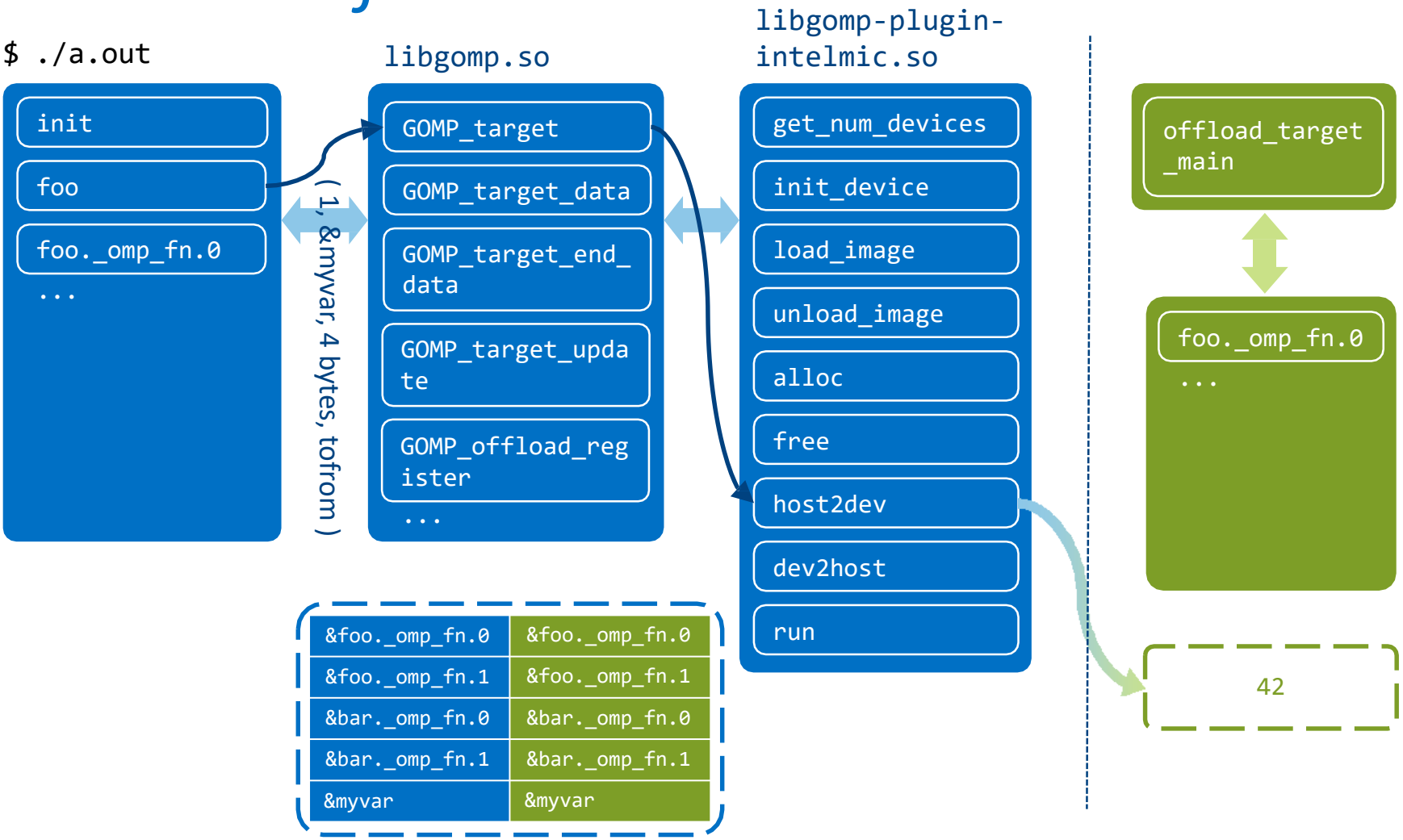
# Implementation in GCC
# Data transfer



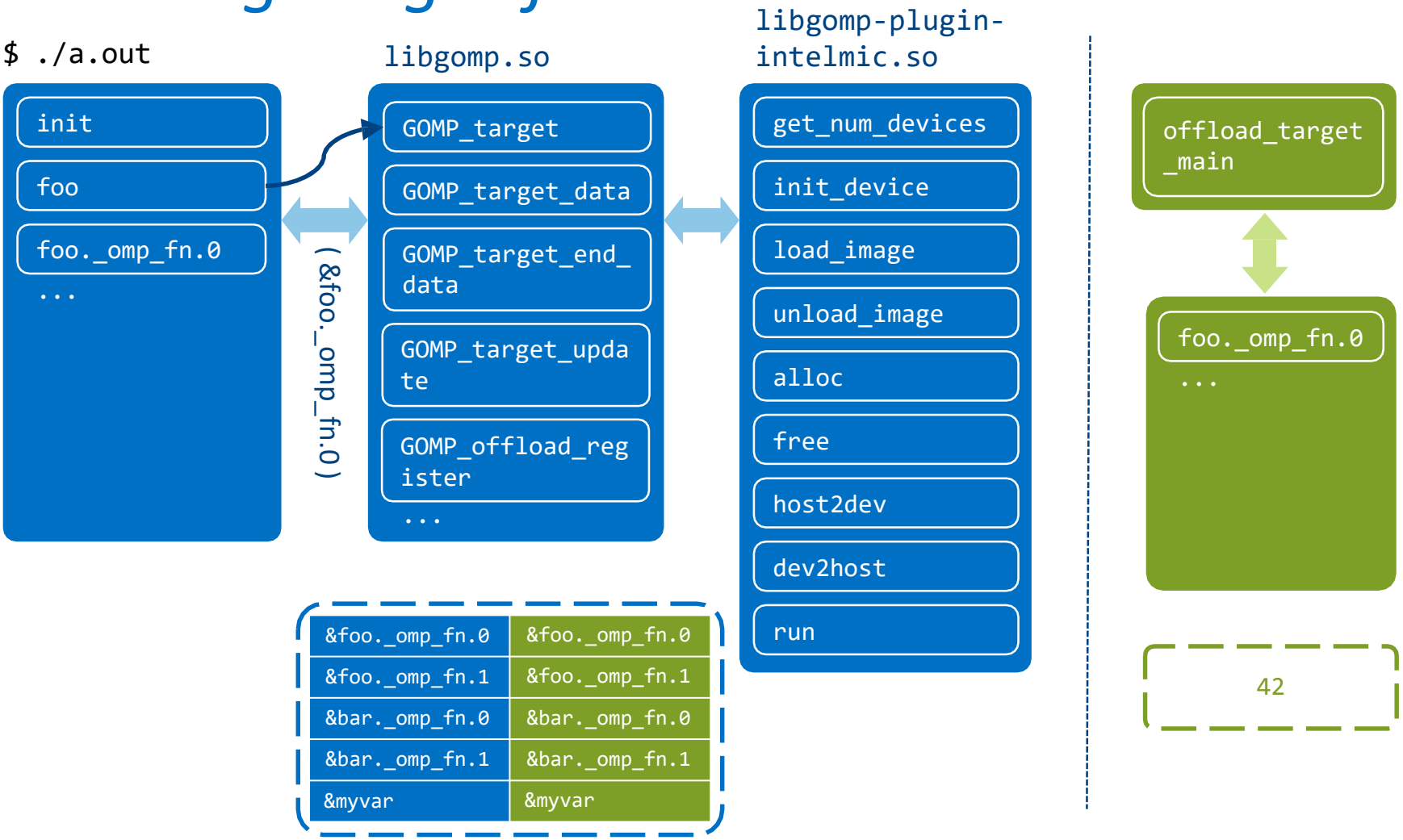
# Implementation in GCC
# Data transfer



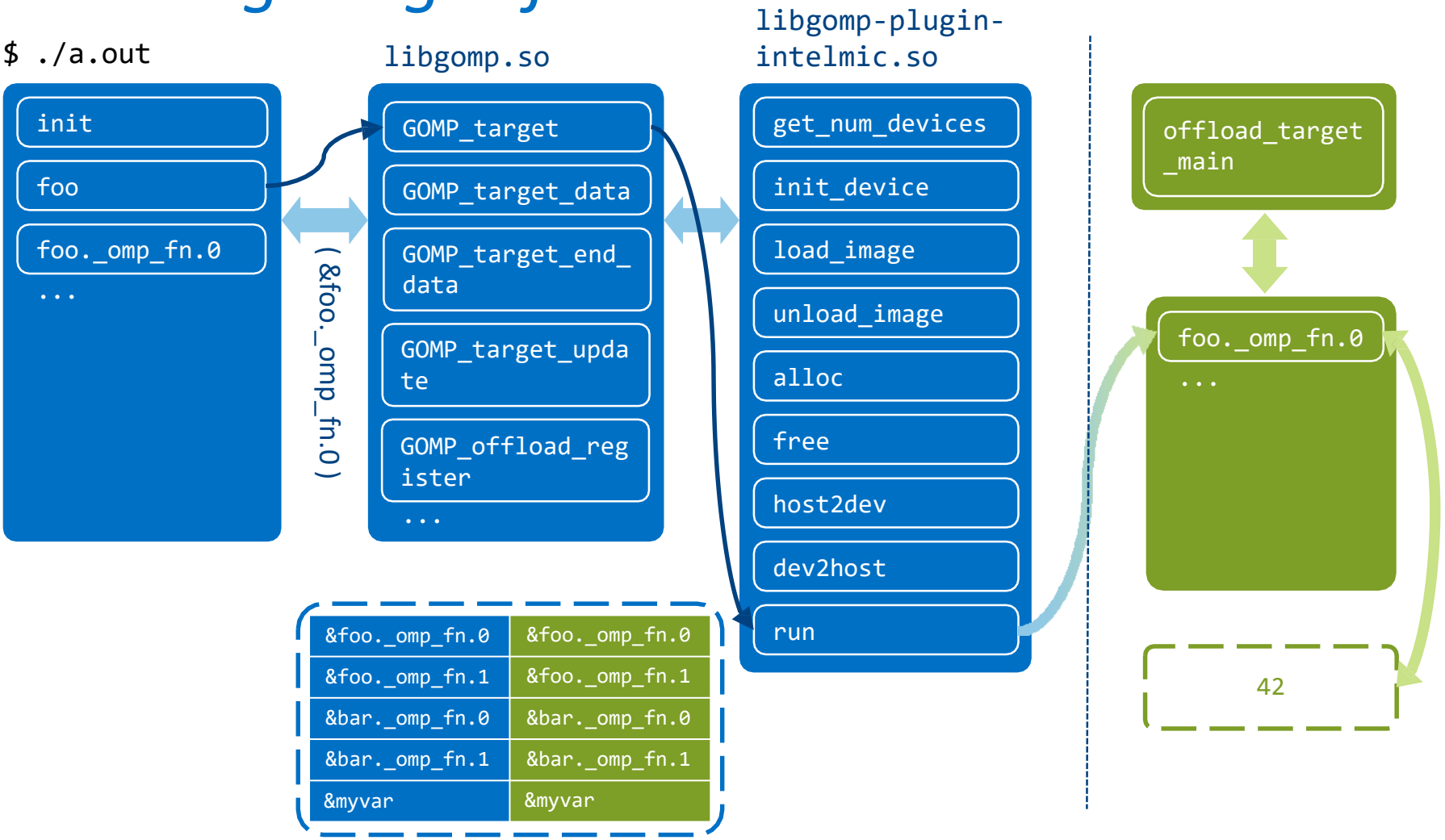
# Implementation in GCC
# Data transfer



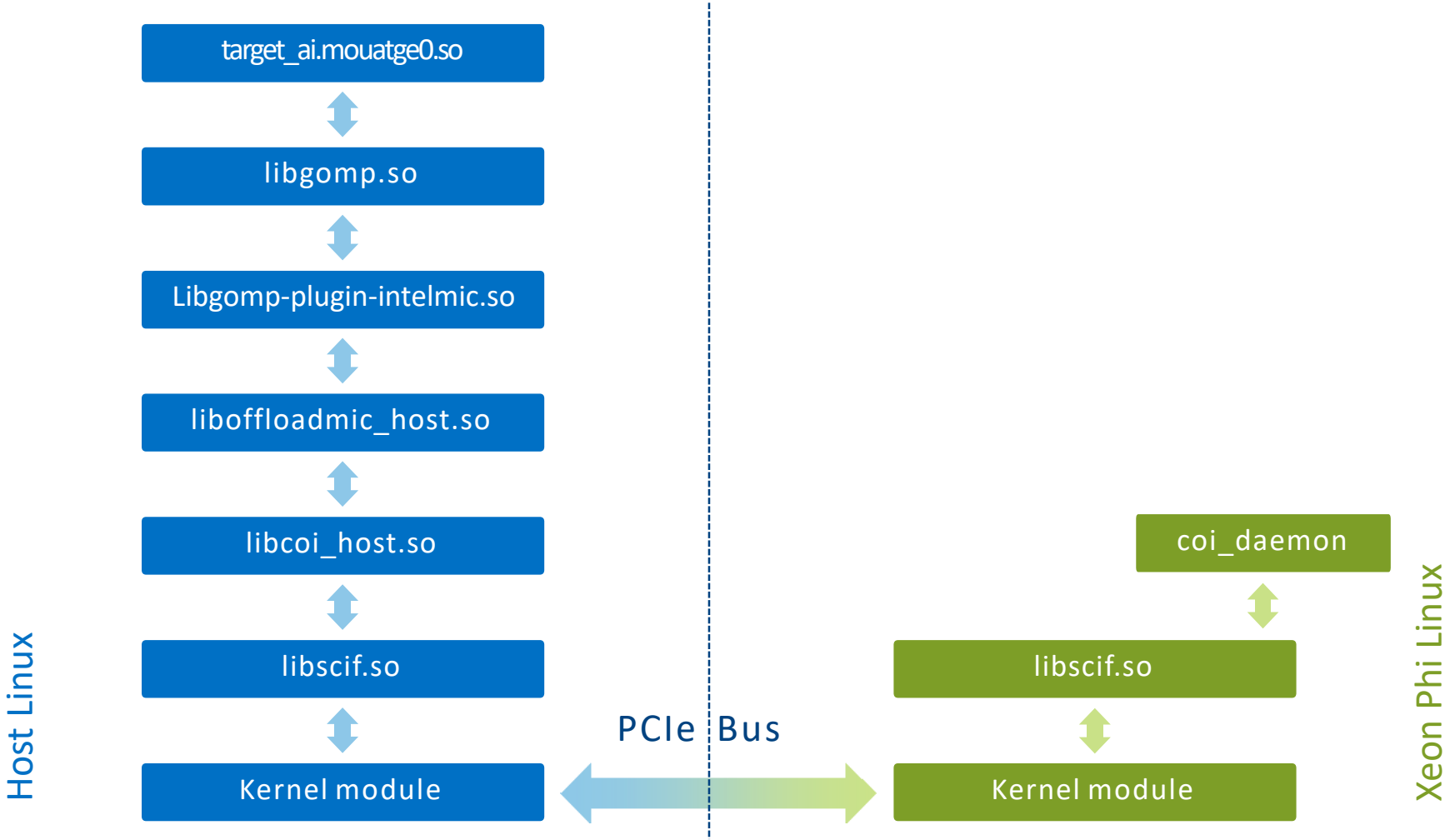
# Implementation in GCC
# Running target funcs



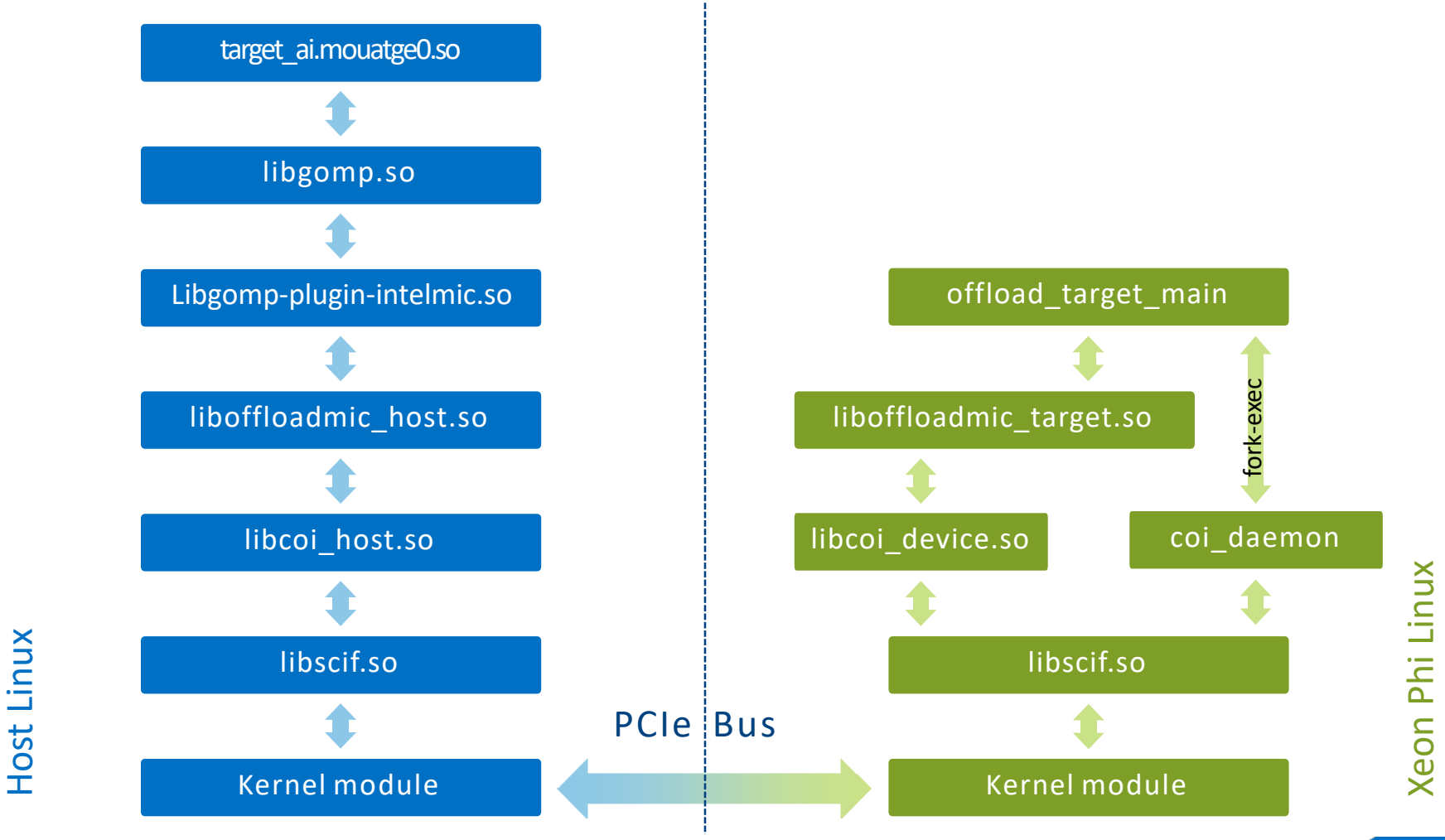
# Implementation in GCC
# Running target funcs



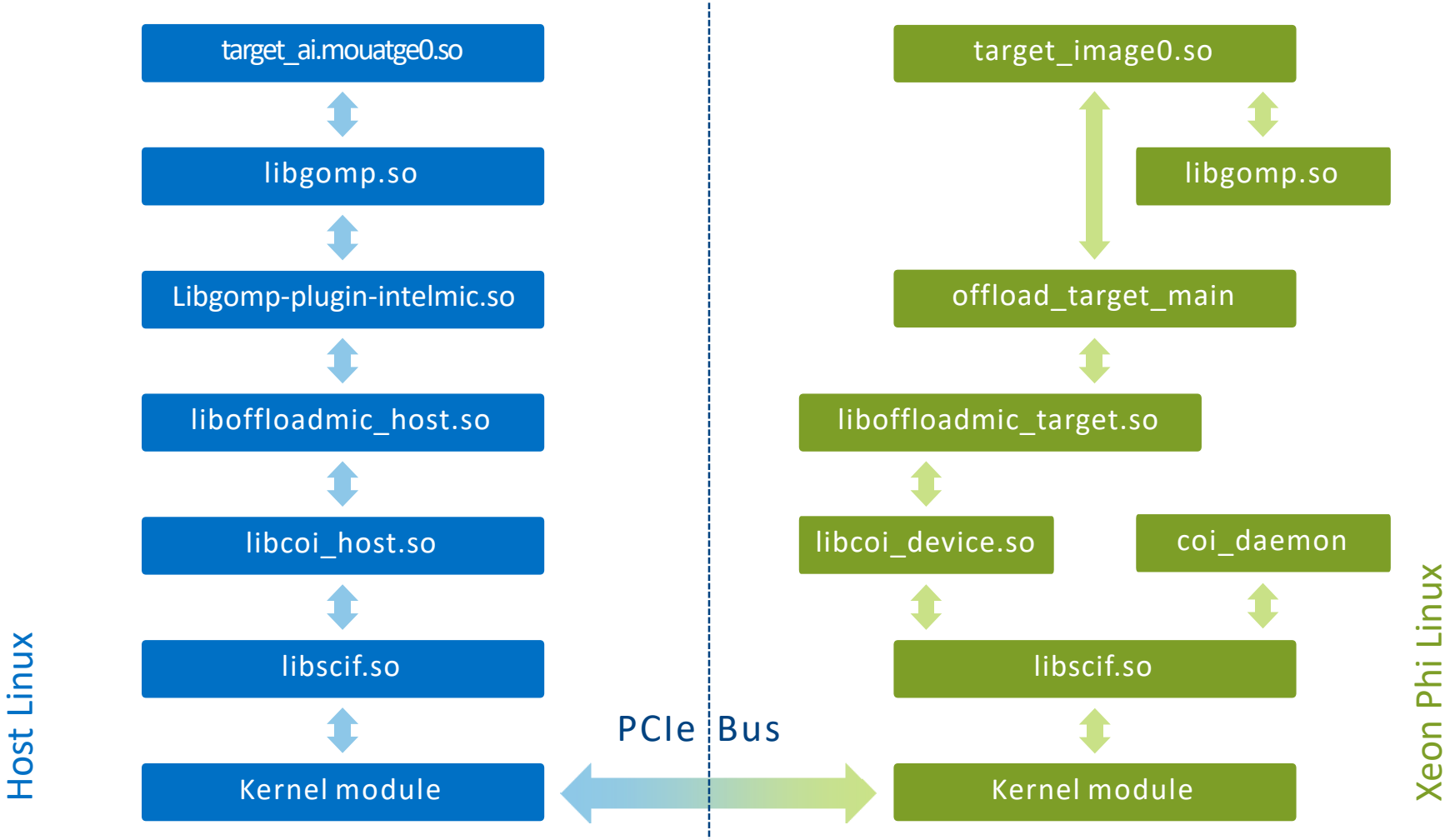
# Implementation in GCC Execution on Intel MIC



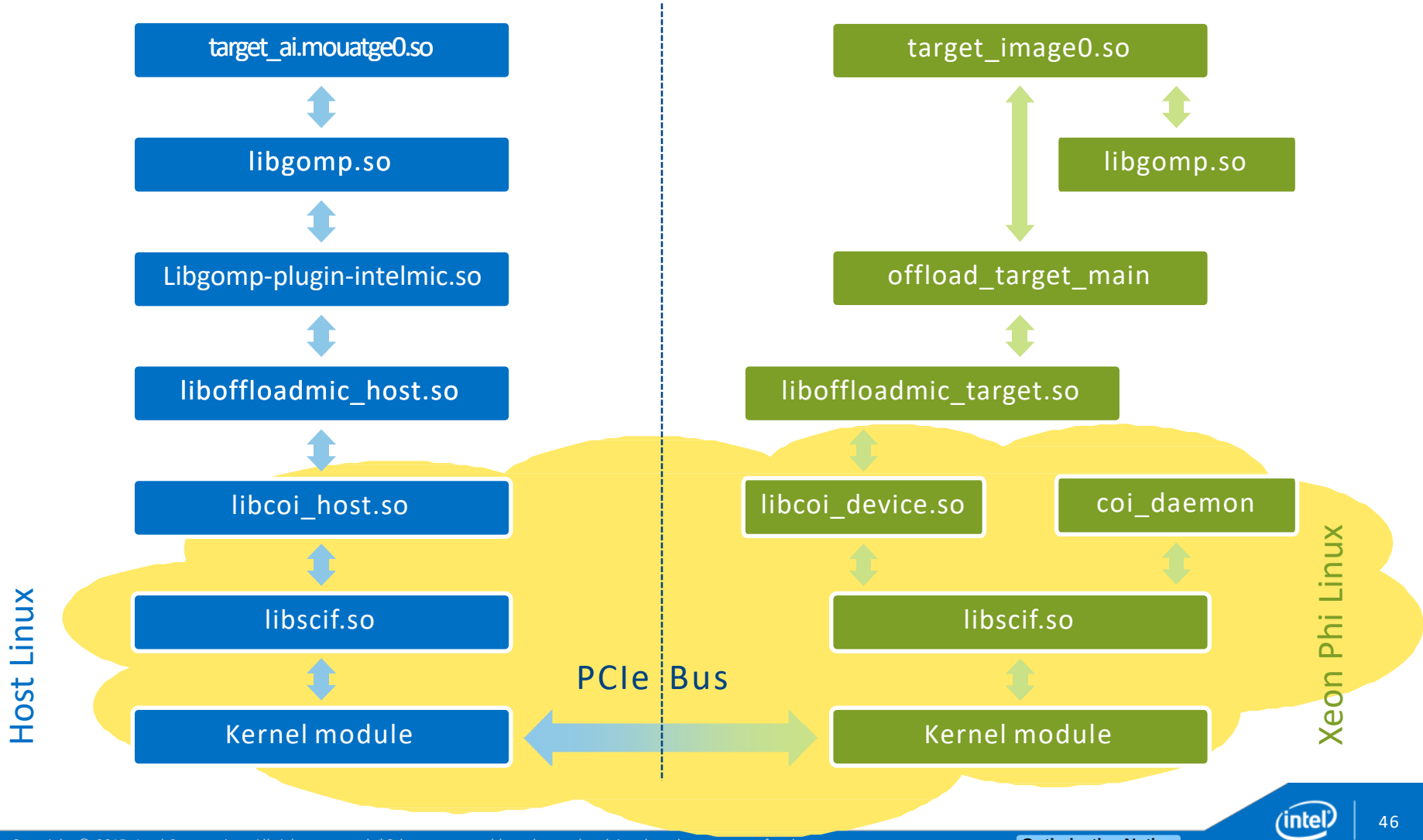
# Implementation in GCC Execution on Intel MIC



# Implementation in GCC Execution on Intel MIC



# Implementation in GCC Execution on Intel MIC

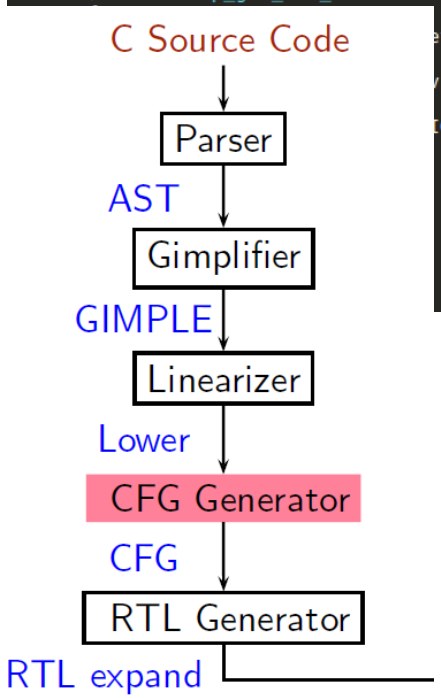


# OpenMP target Expansion

```
19  #include <omp.h>
20  #include <stdlib.h>
21  #include <stdio.h>
22  #include <stdint.h>
23  #include <time.h>
24  #include <hero-targ
25
26  struct timespec start, stop;
27  double start_ns, stop_ns, exe_time;
28
29  #pragma omp declare target
30  void helloworld ()
31  {
32      #pragma omp parallel
33      printf("Hello World, I am thread %d of %d
            omp_get_num_threads());
```

```
et

v[])

IGPULP_MEMCPY);
```

<CROSS-COMPILE>-gcc … -fopenmp -fdump-tree-all ….c




```
hero-vm@ubuntu:~/hero-sdk/hero-openmp-examples/helloworld$ ls
helloworld                          helloworld.c.091t.ccp2              helloworld.c.130t.lim2
helloworld.c                        helloworld.c.092t.post_ipa_warn1    helloworld.c.131t.crited1
helloworld.c.001t.tu                helloworld.c.093t.cunrolli          helloworld.c.133t.pre
helloworld.c.002t.class             helloworld.c.094t.backprop          helloworld.c.134t.sink
helloworld.c.003t.original          helloworld.c.095t.phiprop           helloworld.c.138t.dce4
helloworld.c.004t.gimple            helloworld.c.096t.forwprop2         helloworld.c.139t.fix_loops
helloworld.c.006t.omplower          helloworld.c.097t.objsz2            helloworld.c.167t.no_loop
helloworld.c.007t.lower             helloworld.c.098t.alias             helloworld.c.168t.slp2
helloworld.c.010t.eh                helloworld.c.099t.retslot           helloworld.c.170t.veclower21
helloworld.c.011t.cfg               helloworld.c.100t.fre3              helloworld.c.172t.printf-return-value2
helloworld.c.012t.ompexp            helloworld.c.101t.mergephi2         helloworld.c.173t.reassoc2
helloworld.c.019t.fixup_cfg1        helloworld.c.102t.thread1           helloworld.c.174t.slsr
helloworld.c.020t.ssa               helloworld.c.103t.vrp1              helloworld.c.175t.split-paths
helloworld.c.022t.nothrow           helloworld.c.105t.dce2              helloworld.c.177t.thread3
helloworld.c.027t.fixup_cfg3        helloworld.c.106t.stdarg            helloworld.c.178t.dom3
helloworld.c.028t.inline_param1     helloworld.c.107t.cdce              helloworld.c.179t.strlen
helloworld.c.029t.einline           helloworld.c.108t.cselim            helloworld.c.180t.thread4
helloworld.c.030t.early_optimizations helloworld.c.109t.copyprop1       helloworld.c.181t.vrp2
helloworld.c.031t.objsz1            helloworld.c.110t.ifcombine         helloworld.c.182t.phicprop2
helloworld.c.032t.ccp1              helloworld.c.111t.mergephi3         helloworld.c.183t.dse3
helloworld.c.033t.forwprop1         helloworld.c.112t.phiopt1           helloworld.c.184t.cddce3
helloworld.c.034t.ethread           helloworld.c.113t.tailr2            helloworld.c.185t.forwprop4
helloworld.c.035t.esra              helloworld.c.114t.ch2               helloworld.c.186t.phiopt3
helloworld.c.036t.ealias            helloworld.c.115t.cplxlower1        helloworld.c.187t.fab1
helloworld.c.037t.fre1              helloworld.c.116t.sra               helloworld.c.188t.widening_mul
helloworld.c.038t.evrp              helloworld.c.117t.thread2           helloworld.c.189t.store-merging
helloworld.c.039t.mergephi1         helloworld.c.118t.dom2              helloworld.c.190t.tailc
helloworld.c.040t.dse1              helloworld.c.119t.isolate-paths     helloworld.c.191t.dce7
helloworld.c.041t.cddce1            helloworld.c.120t.phicprop1         helloworld.c.192t.crited2
helloworld.c.042t.eipa_sra          helloworld.c.121t.dse2              helloworld.c.193t.uninit1
helloworld.c.043t.tailr1            helloworld.c.122t.reassoc1          helloworld.c.194t.uncprop1
helloworld.c.044t.switchconv        helloworld.c.123t.dce3              helloworld.c.195t.local-pure-const2
helloworld.c.046t.profile_estimate  helloworld.c.124t.forwprop3         helloworld.c.226t.nrv
helloworld.c.047t.local-pure-const1 helloworld.c.125t.phiopt2           helloworld.c.227t.optimized
helloworld.c.048t.fnsplit           helloworld.c.126t.ccp3              helloworld.c.308t.statistics
helloworld.c.049t.release_ssa       helloworld.c.127t.sincos            helloworld.o
helloworld.c.050t.inline_param2     helloworld.c.128t.bswap             Makefile
helloworld.c.086t.fixup_cfg4        helloworld.c.129t.laddress
```

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# OpenMP target – Howto Manage multiple ISA? Taking a look to the device optimizations

```
helloworld.c.031t.objsz1              helloworld.c.110t.ifcombine       helloworld.c.182t.phicprop2
helloworld.c.032t.ccp1                helloworld.c.111t.mergephi3       helloworld.c.183t.dse3
helloworld.c.033t.forwprop1           helloworld.c.112t.phiopt1         helloworld.c.184t.cddce3
helloworld.c.034t.ethread             helloworld.c.113t.tailr2          helloworld.c.185t.forwprop4
helloworld.c.035t.esra                helloworld.c.114t.ch2             helloworld.c.186t.phiopt3
helloworld.c.036t.ealias              helloworld.c.115t.cplxlower1      helloworld.c.187t.fab1
helloworld.c.037t.fre1                helloworld.c.116t.sra             helloworld.c.188t.widening_mul
helloworld.c.038t.evrp                helloworld.c.117t.thread2         helloworld.c.189t.store-merging
helloworld.c.039t.mergephi1           helloworld.c.118t.dom2            helloworld.c.190t.tailc
helloworld.c.040t.dse1                helloworld.c.119t.isolate-paths   helloworld.c.191t.dce7
helloworld.c.041t.cddce1              helloworld.c.120t.phicprop1       helloworld.c.192t.crited2
helloworld.c.042t.eipa_sra            helloworld.c.121t.dse2            helloworld.c.193t.uninit1
helloworld.c.043t.tailr1              helloworld.c.122t.reassoc1        helloworld.c.194t.uncprop1
helloworld.c.044t.switchconv          helloworld.c.123t.dce3            helloworld.c.195t.local-pure-const2
helloworld.c.046t.profile_estimate    helloworld.c.124t.forwprop3       helloworld.c.226t.nrv
helloworld.c.047t.local-pure-const1   helloworld.c.125t.phiopt2         helloworld.c.227t.optimized
helloworld.c.048t.fnsplit             helloworld.c.126t.ccp3            helloworld.c.308t.statistics
helloworld.c.049t.release_ssa         helloworld.c.127t.sincos          helloworld.o
helloworld.c.050t.inline_param2       helloworld.c.128t.bswap           Makefile
helloworld.c.086t.fixup_cfg4          helloworld.c.129t.laddress
hero-vm@ubuntu:~/hero-sdk/hero-openmp-examples/helloworld$ ls /tmp/
config-err-3qwoUH                     helloworld.o.130t.lim2
gnome-software-ZBF21Z                 helloworld.o.131t.crited1
helloworld.o.046t.profile_estimate    helloworld.o.133t.pre
helloworld.o.086t.fixup_cfg4          helloworld.o.134t.sink
helloworld.o.090t.omptargetlink       helloworld.o.138t.dce4
helloworld.o.091t.ccp2                helloworld.o.139t.fix_loops
helloworld.o.093t.cunrolli            helloworld.o.167t.no_loop
helloworld.o.094t.backprop            helloworld.o.168t.slp2
helloworld.o.095t.phiprop             helloworld.o.170t.veclower21
helloworld.o.096t.forwprop2           helloworld.o.172t.printf-return-value2
helloworld.o.097t.objsz2              helloworld.o.173t.reassoc2
helloworld.o.098t.alias               helloworld.o.174t.slsr
helloworld.o.099t.retslot             helloworld.o.175t.split-paths
helloworld.o.100t.fre3                helloworld.o.177t.thread3
helloworld.o.101t.mergephi2           helloworld.o.178t.dom3
helloworld.o.102t.thread1             helloworld.o.179t.strlen
helloworld.o.103t.vrp1                helloworld.o.180t.thread4
helloworld.o.105t.dce2                helloworld.o.181t.vrp2
helloworld.o.106t.stdarg              helloworld.o.182t.phicprop2
helloworld.o.107t.cdce                helloworld.o.183t.dse3
helloworld.o.108t.cselim              helloworld.o.184t.cddce3
helloworld.o.109t.copyprop1           helloworld.o.185t.forwprop4
helloworld.o.110t.ifcombine           helloworld.o.186t.phiopt3
helloworld.o.111t.mergephi3           helloworld.o.187t.fab1
helloworld.o.112t.phiopt1             helloworld.o.188t.widening_mul
helloworld.o.113t.tailr2              helloworld.o.189t.store-merging
helloworld.o.114t.ch2                 helloworld.o.190t.tailc
helloworld.o.115t.cplxlower1          helloworld.o.191t.dce7
helloworld.o.116t.sra                 helloworld.o.192t.crited2
helloworld.o.117t.thread2             helloworld.o.193t.uninit1
helloworld.o.118t.dom2                helloworld.o.194t.uncprop1
helloworld.o.119t.isolate-paths       helloworld.o.195t.local-pure-const2
helloworld.o.120t.phicprop1           helloworld.o.226t.nrv
helloworld.o.121t.dse2                helloworld.o.227t.optimized
helloworld.o.122t.reassoc1            helloworld.o.308t.statistics
helloworld.o.123t.dce3                systemd-private-4cb8d36101d34251aa4f3e17b64626df-colord.service-7aLWSC
helloworld.o.124t.forwprop3           systemd-private-4cb8d36101d34251aa4f3e17b64626df-fwupd.service-srRDBM
helloworld.o.125t.phiopt2             systemd-private-4cb8d36101d34251aa4f3e17b64626df-rtkit-daemon.service-a3XxPp
helloworld.o.126t.ccp3                systemd-private-4cb8d36101d34251aa4f3e17b64626df-systemd-timesyncd.service-kZqVWn
helloworld.o.127t.sincos              unity_support_test.1
helloworld.o.128t.bswap               VMwareDnD
helloworld.o.129t.laddress            vmware-root
```

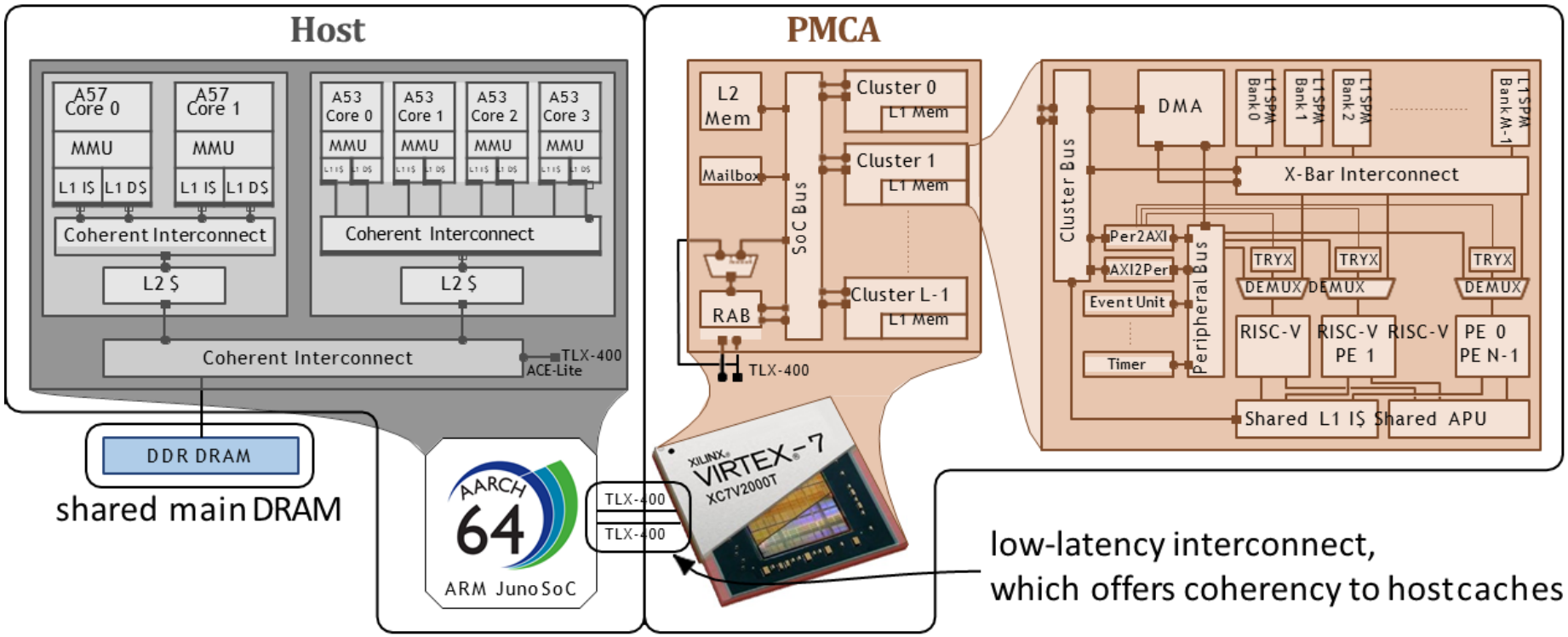
Where are all the other steps???

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# HERO's Hardware Architecture

industry-standard, hard-macro ARM Cortex-A Host processor

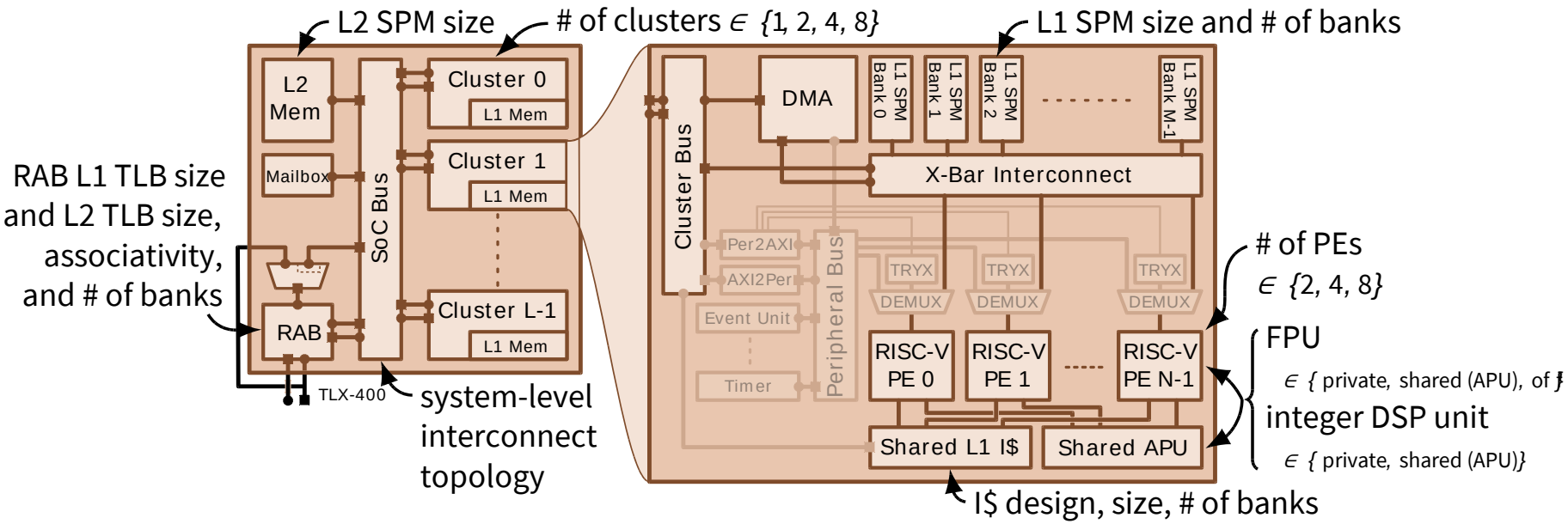
scalable, configurable, modifiable FPGA implementation of a silicon-proven, cluster-based PMCA with RISC-V PEs




HERO's hardware, as implemented on the Juno ADP.

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# HERO is modifiable and expandable



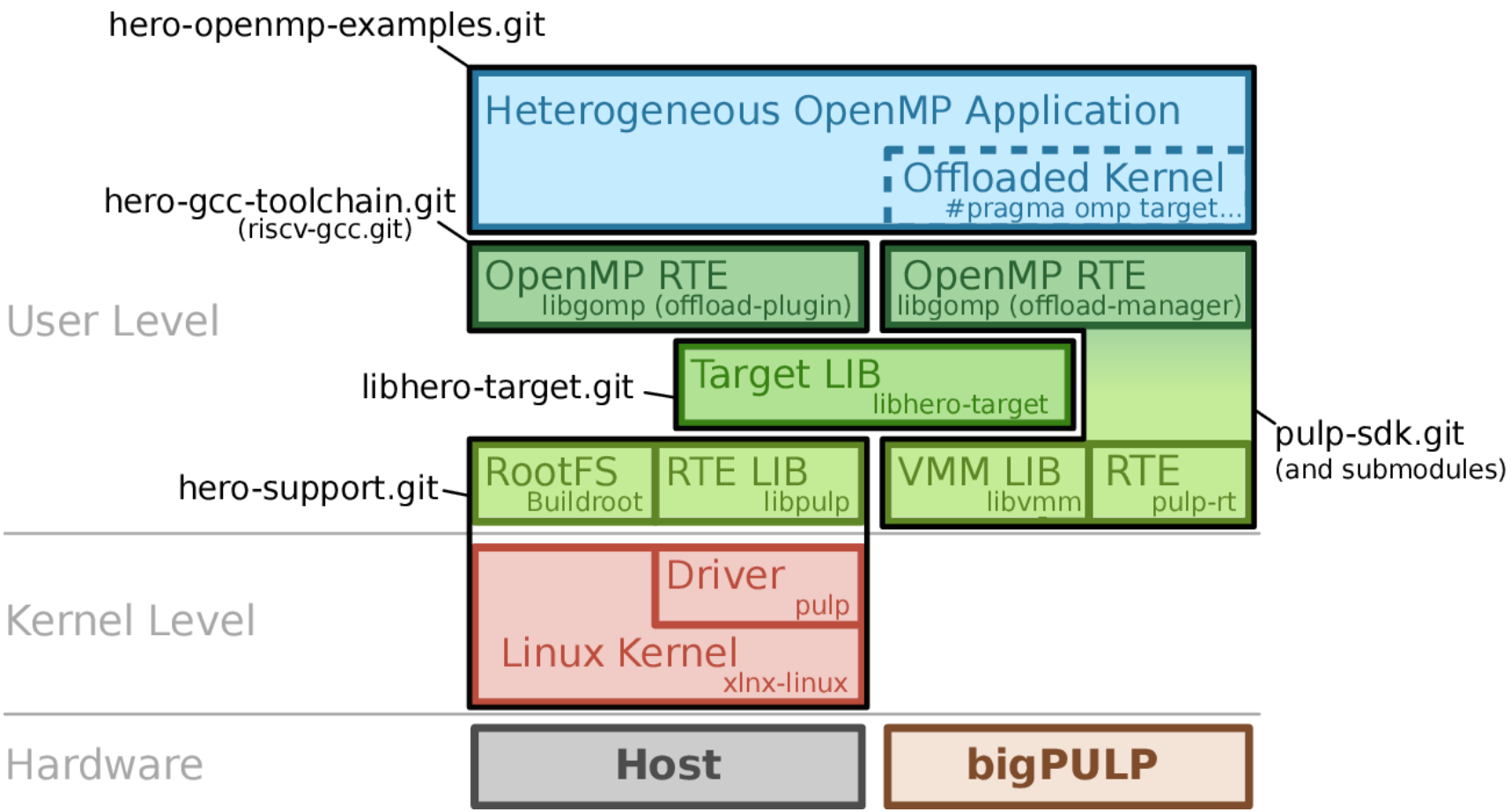

- Up to 8 clusters, each with 8 cores
- All components are open source and written in System Verilog
- Standard interfaces (mostly AXI)
- New components can easily be added to the memory map

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# Supported Platforms and Configurations

| Property | ARM Juno (with a Xilinx Virtex-7 2000T) | Xilinx Zynq UltraScale+ ZU9 | Xilinx Zynq ZC706 |
|---|---|---|---|
| Host CPU | 64-bit ARMv8 big.LITTLE | 64-bit ARMv8 quad-core A53 | 32-bit ARMv7 dual-core A9 |
| Shared main memory | 8 GiB DDR3L | 2 GiB DDR4 | 1 GiB DDR3 |
| PMCA clock frequency | **31 MHz** | **145 MHz** | **57 MHz** |
| # of RISC-V PEs | **64 in 8 clusters** | **8 in 1 cluster** | **8 in 1 cluster** |
| Integer DSP unit L1 SPM | private per PE 256 KiB in 16 banks | | |
| Instruction cache | 8 KiB in 8 single-ported banks | 4 KiB in 4 multi-ported banks | |
| Slices used by clusters | 80% | 48% | 65% |
| Slices used by infrastructure | 7% | 10% | 12% |
| BRAMs used by clusters | 89% | 42% | 70% |
| BRAMs used by infrastructure | 6% | 8% | 13% |
| Price | 20 000 $ | 2500 $ | 900 $ |

# Not only open hardware. The HERO-SDK




PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

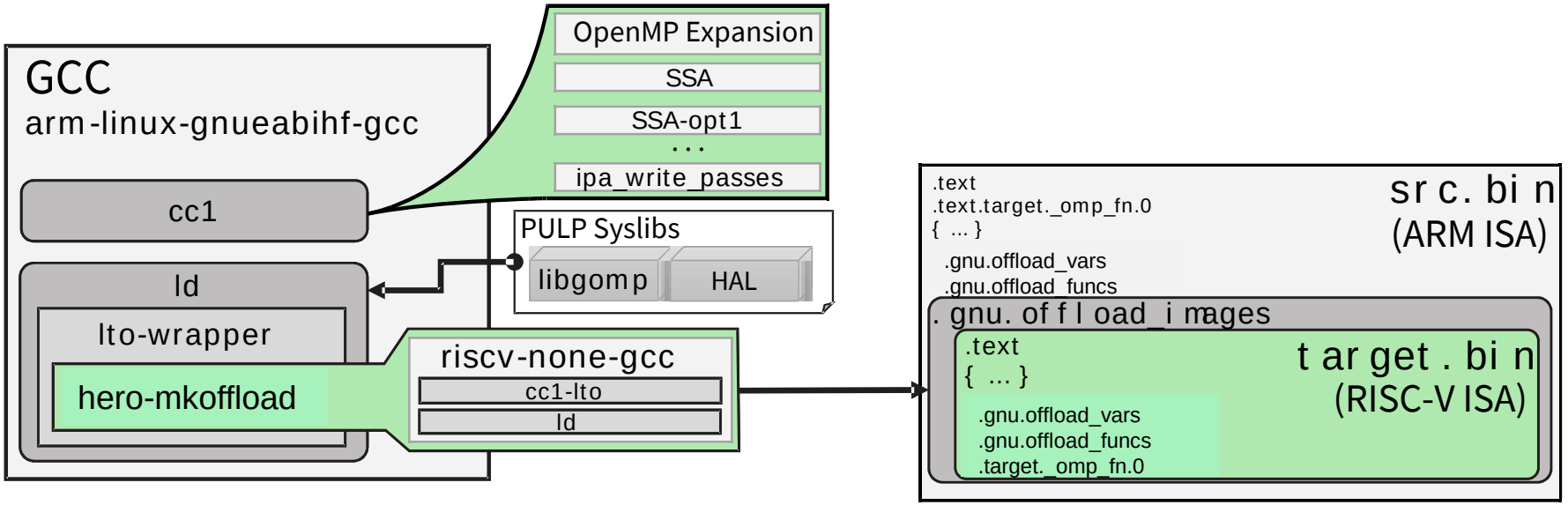
# HERO GNU GCC 7.1 - Extensions

- **Added HERO as target accelerator**
  - Enabled RISC-V back-end as OpenMP4 accelerator supported ISA
  - Created *ad-hoc lto-wrapped* linker tool for HERO offloaded region (***pulp-mkoffload***)
  - New libgomp plugin for HERO
- **Enabled *Shared Virtual Memory* (zero-copy) support for HERO**
  - Added new SSA pass to protect usage of shared mapped variables between the accelerator and the host

**First Not Commercial Architecture with GCC OpenMP offloading support**

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

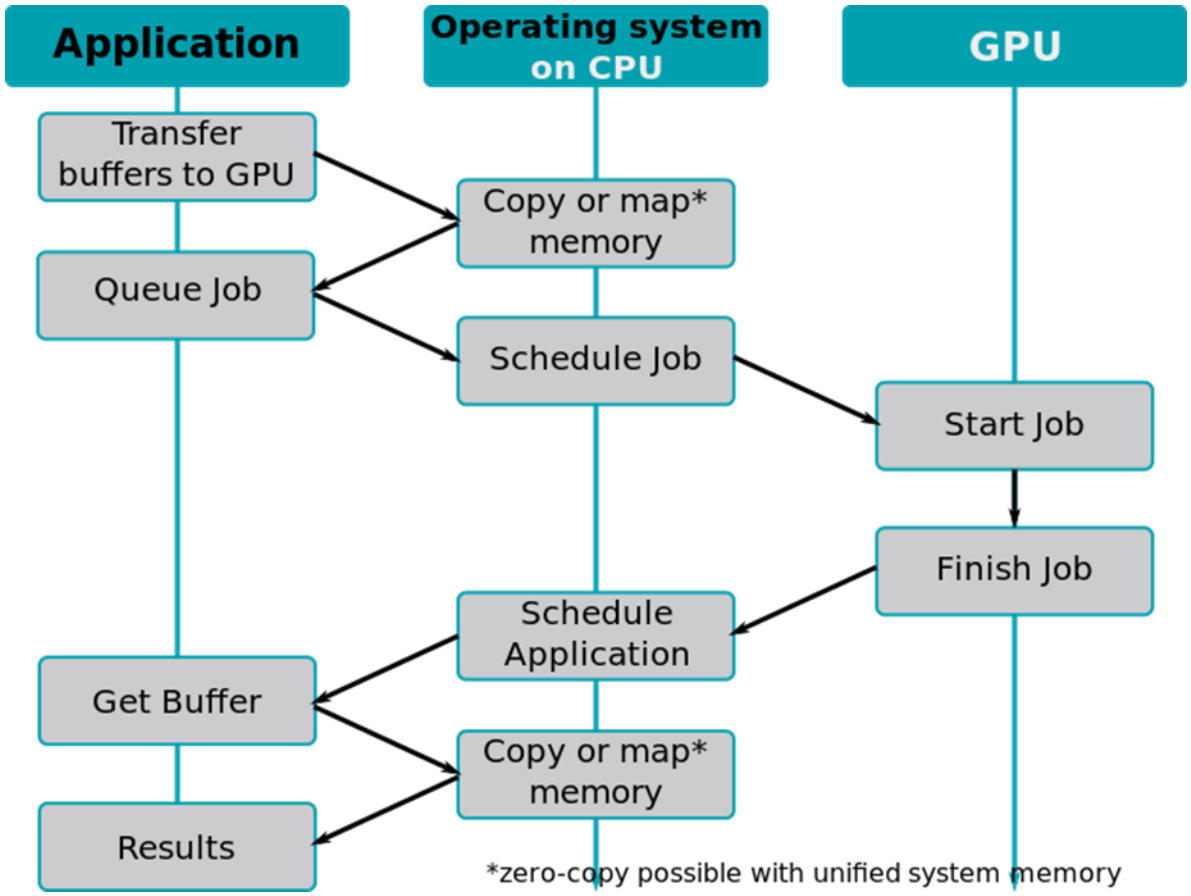
# Heterogeneous cross compilation toolchain of HERO




- OpenMP offloading with the GCC toolchain
  - requires host compiler and a target compiler for each PMCA ISA.
  - Target compiler needs both compiler and runtime extensions (i.e. for libgomp)
- HERO includes the first non-commercial heterogeneous cross compilation toolchain

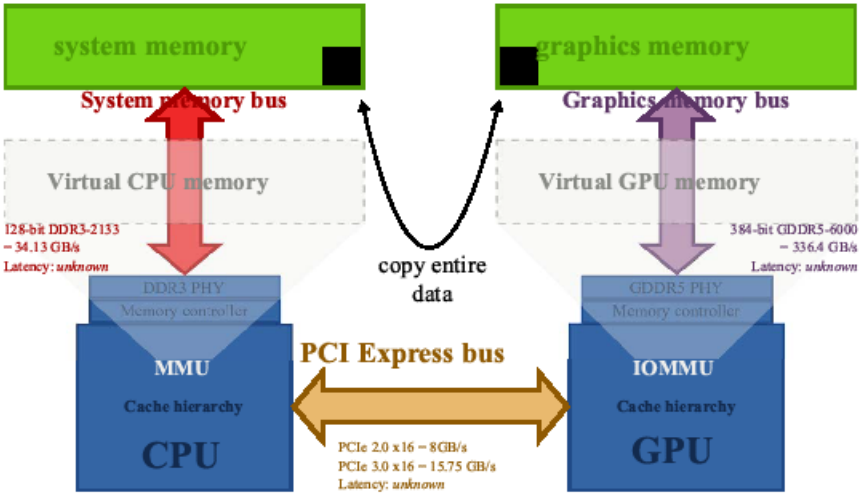
PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

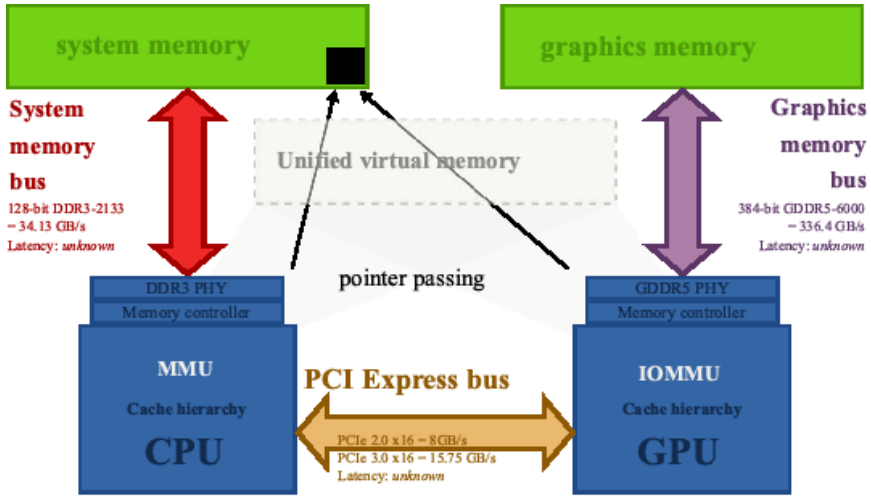
# Heterogeneous Manycores Offloading



# Heterogenous Memory Models

- **Separated Virtual Memories**
  - No Hardware Required
  - No Interference Between Host and Accelerator
  - Explicit Data Copy:
    - Performance Overheads
    - No Pointers Support?
    - Requires Code Modification!
- **Unified Virtual Memories**
  - Hardware Required
  - Interference Between Host and Accelerator (be-aware!)
  - Zero-Copy Enable
    - Performance (Overheads?!)
    - Passing Pointers!!
    - No Code Modification!! (Theoretically)



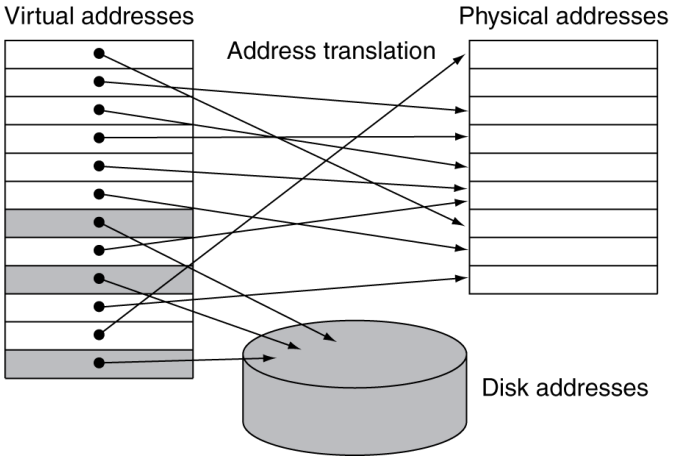

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

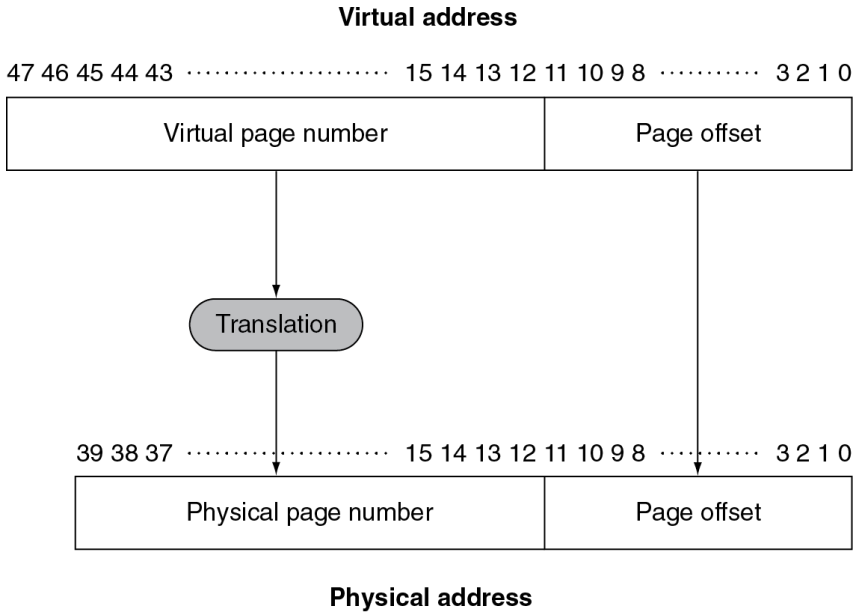
# Virtual Memory (Recall)

- Use main memory as a "cache" for secondary (disk) storage
  - Managed jointly by CPU hardware and the operating system (OS)
- Programs share main memory
  - Each gets a private virtual address space holding its frequently used code and data
  - Protected from other programs
- CPU and OS translate virtual addresses to physical addresses
  - VM "block" is called a page
  - VM translation "miss" is called a page fault

# Address Translation (Recall)

- Fixed-size pages (e.g., 4K)

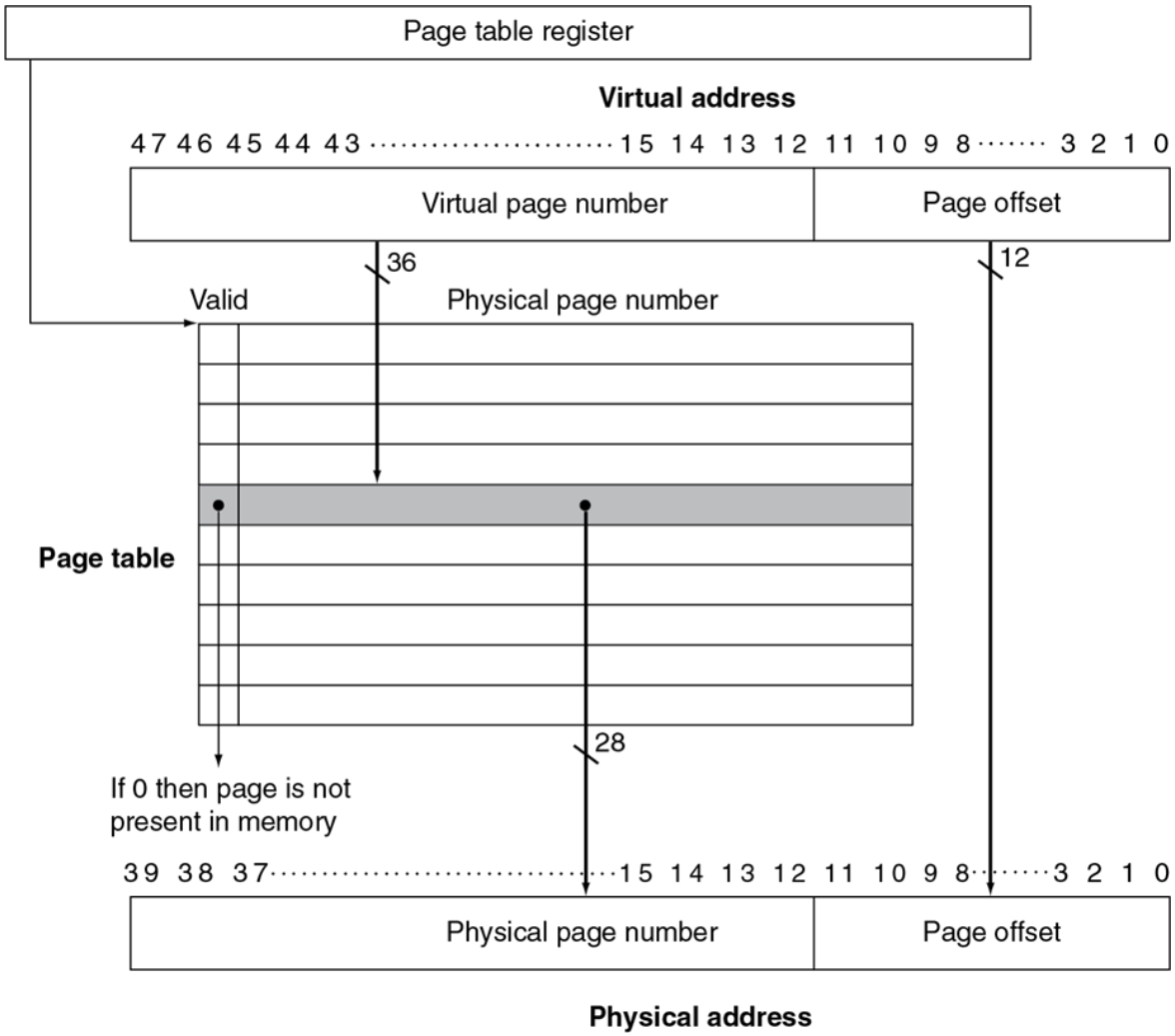


# Page Tables (Recall)

- Stores placement information
  - Array of page table entries, indexed by virtual page number
  - Page table register in CPU points to page table in physical memory
- If page is present in memory
  - PTE stores the physical page number
  - Plus other status bits (referenced, dirty, …)
- If page is not present
  - PTE can refer to location in swap space on disk

# Translation Using a Page Table (Recall)

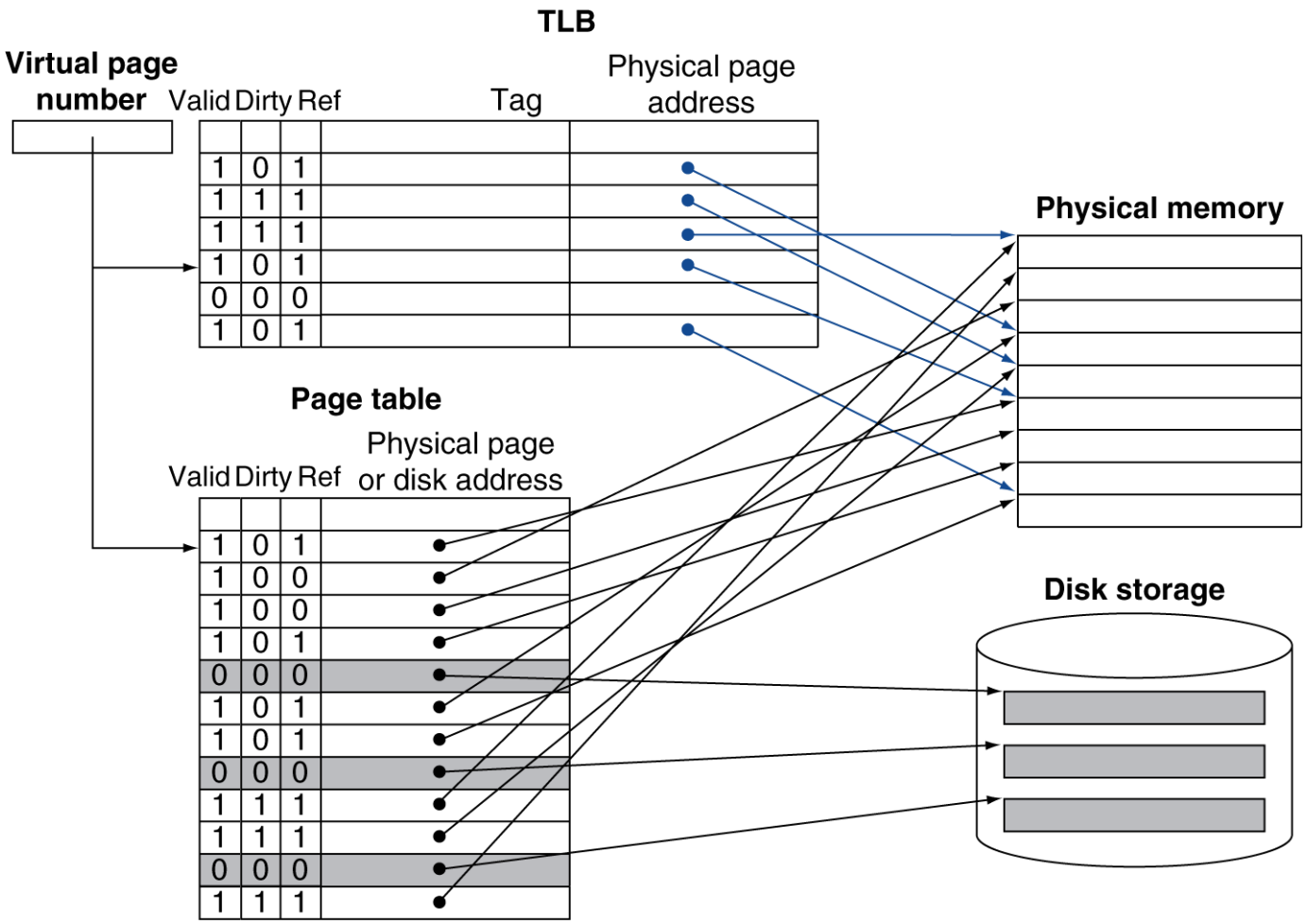


# Fast Translation Using a TLB (Recall)

- Address translation would appear to require extra memory references
  - One to access the PTE
  - Then the actual memory access

- But access to page tables has good locality
  - So use a fast cache of PTEs within the CPU
  - Called a Translation Look-aside Buffer (TLB)
  - Typical: 16–512 PTEs, 0.5–1 cycle for hit, 10–100 cycles for miss, 0.01%–1% miss rate
  - Misses could be handled by hardware or software

# Fast Translation Using a TLB (Recall)

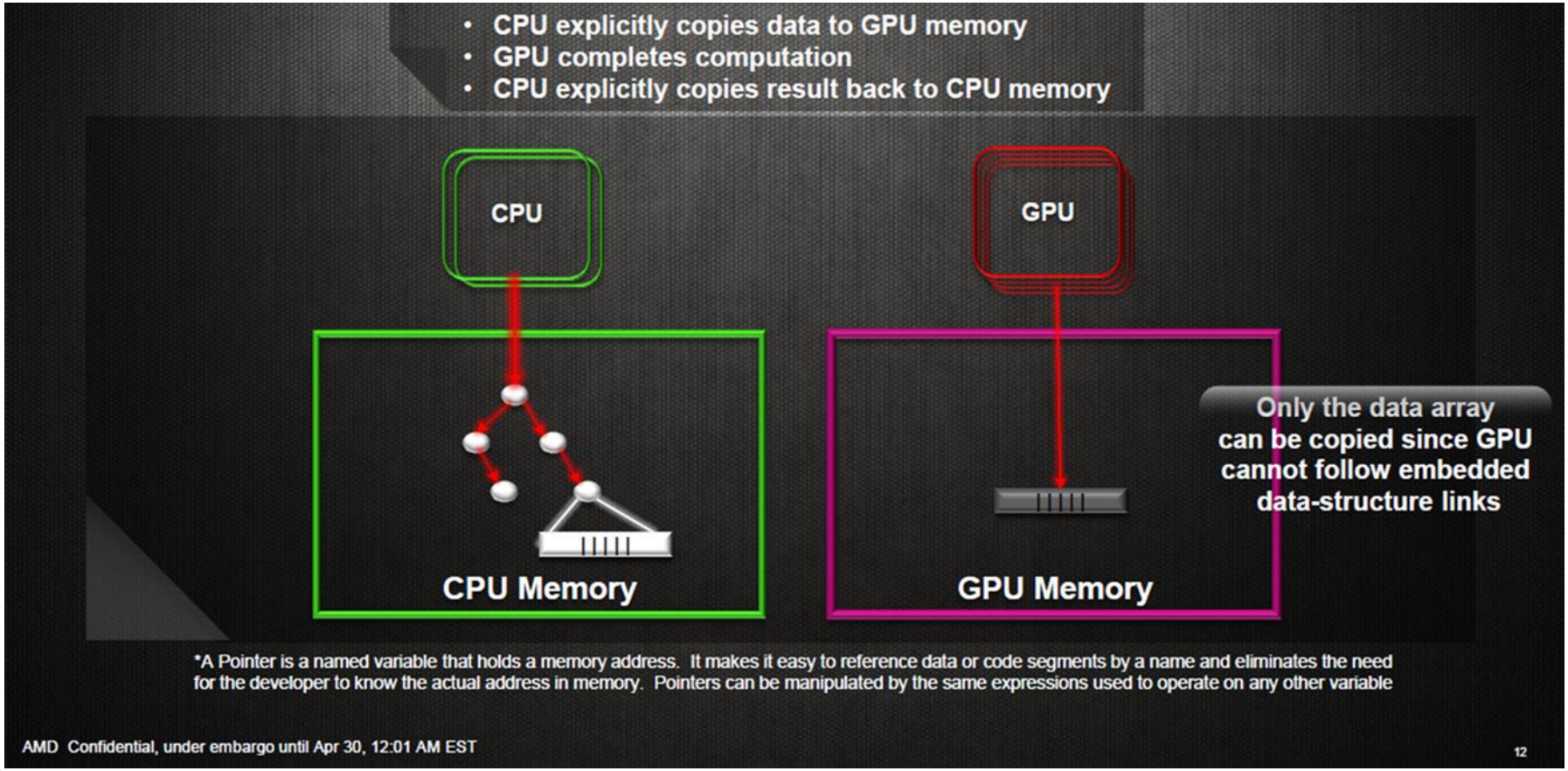



PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# TLB Misses (Recall)

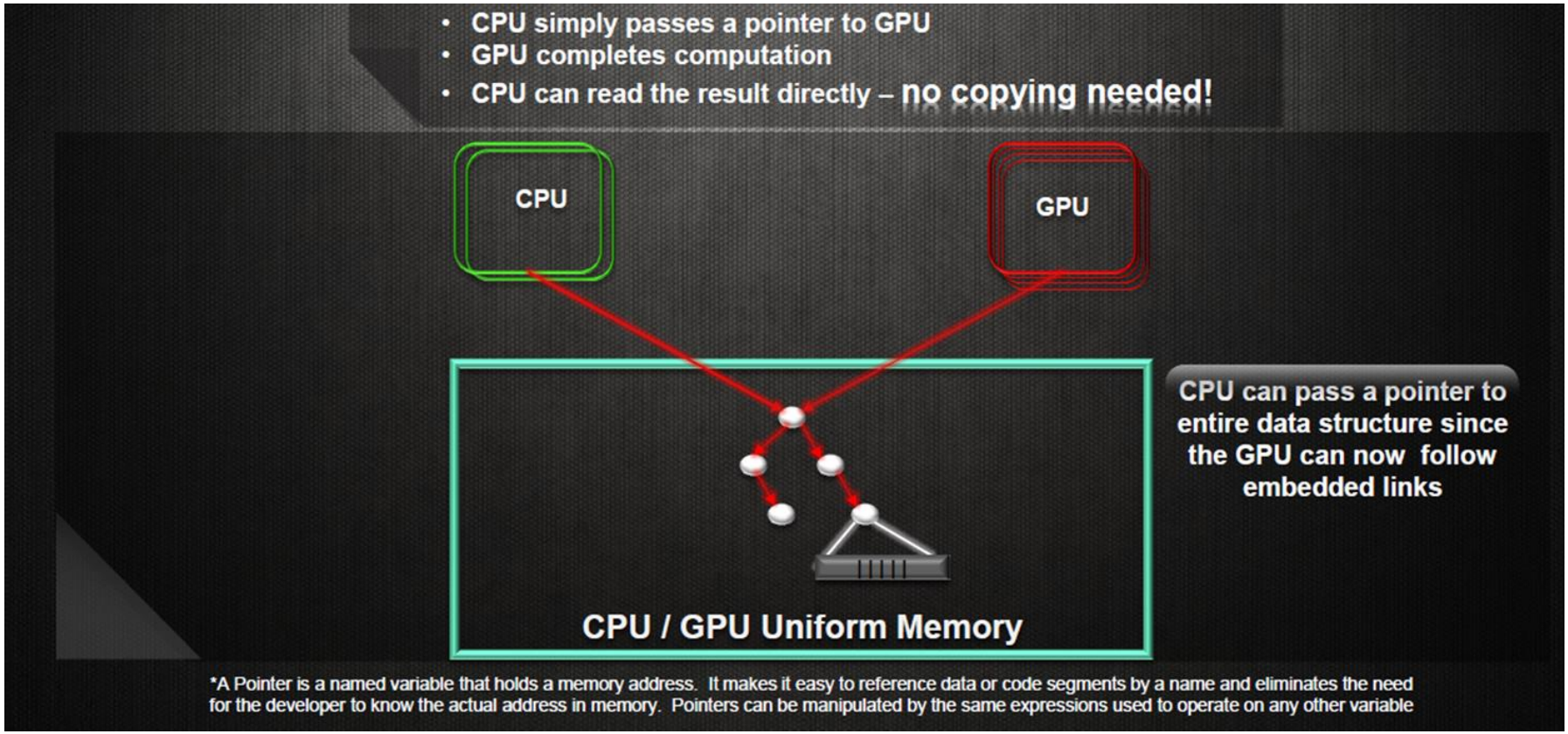
- If page is in memory
  - Load the PTE from memory and retry
  - Could be handled in hardware
    - Can get complex for more complicated page table structures
  - Or in software
    - Raise a special exception, with optimized handler
- If page is not in memory (page fault)
  - OS handles fetching the page and updating the page table
  - Then restart the faulting instruction

# Separated Virtual Memories

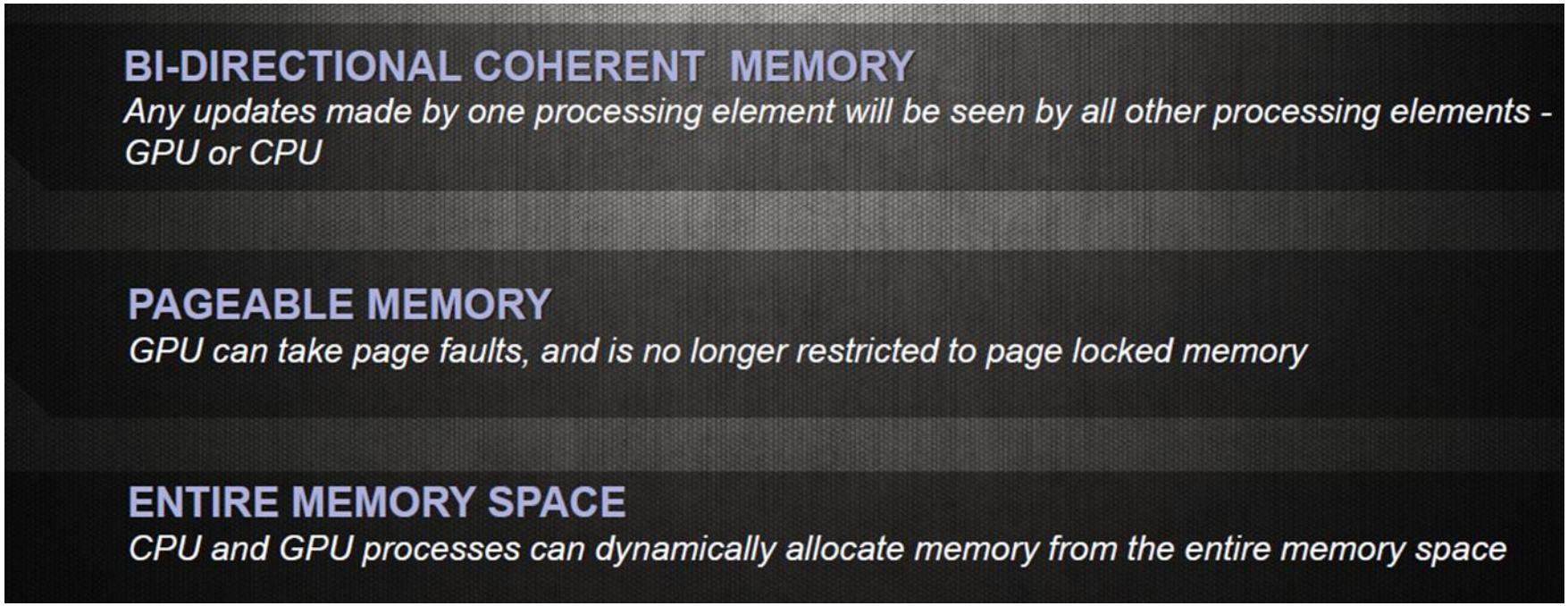



PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

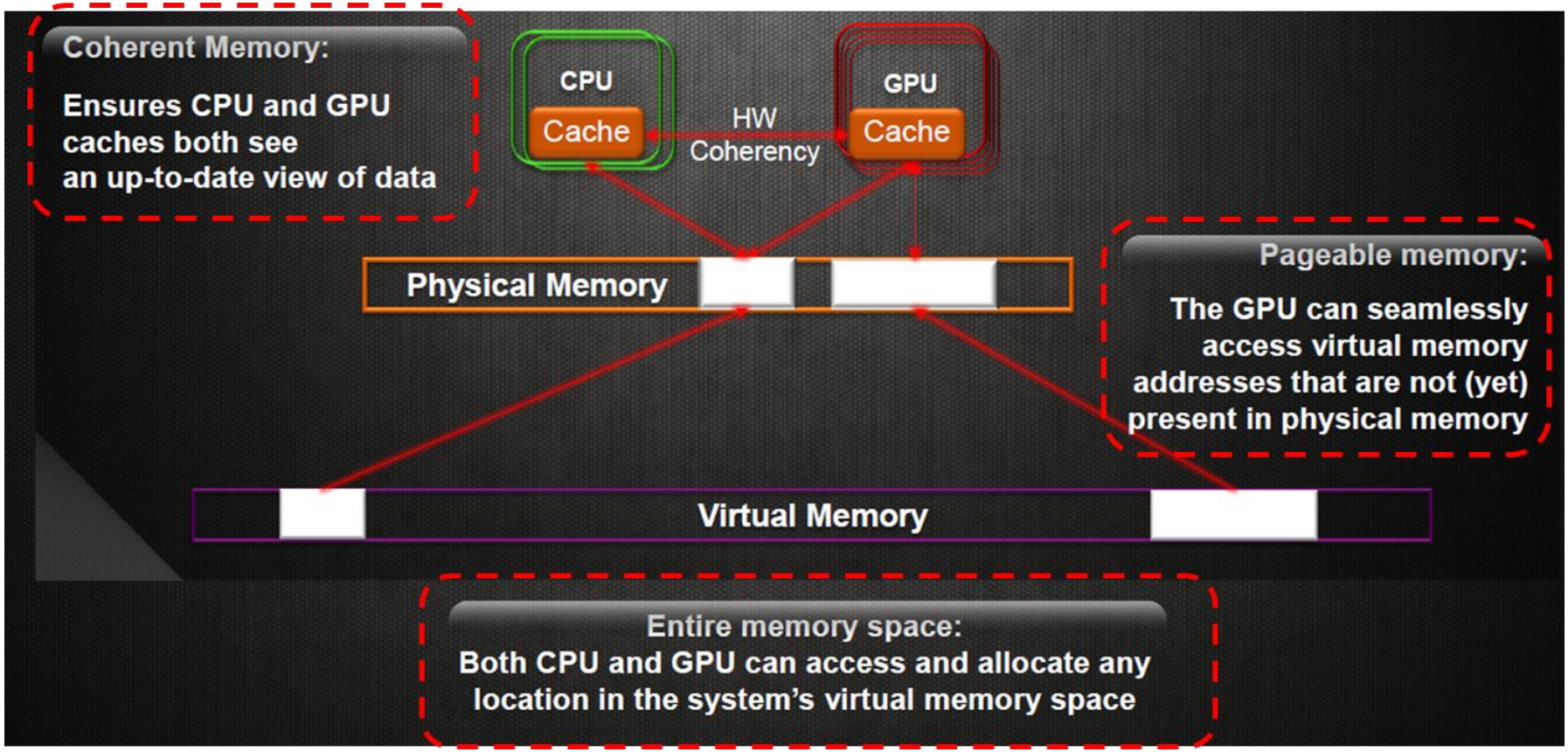
# Unified Virtual Memories (i.e. HSA hUMA)



# Key Hardware Requirement (i.e. HSA hUMA)

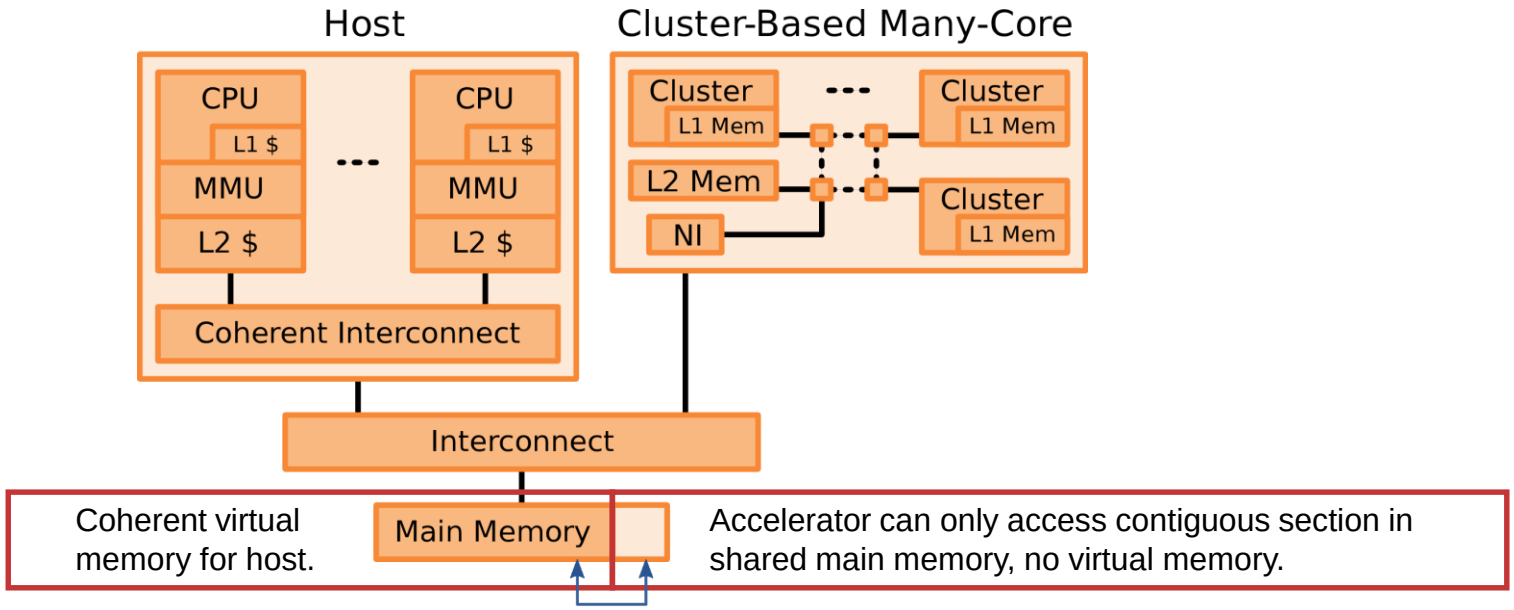


# Key Hardware Requirement (i.e. HSA hUMA) - 2



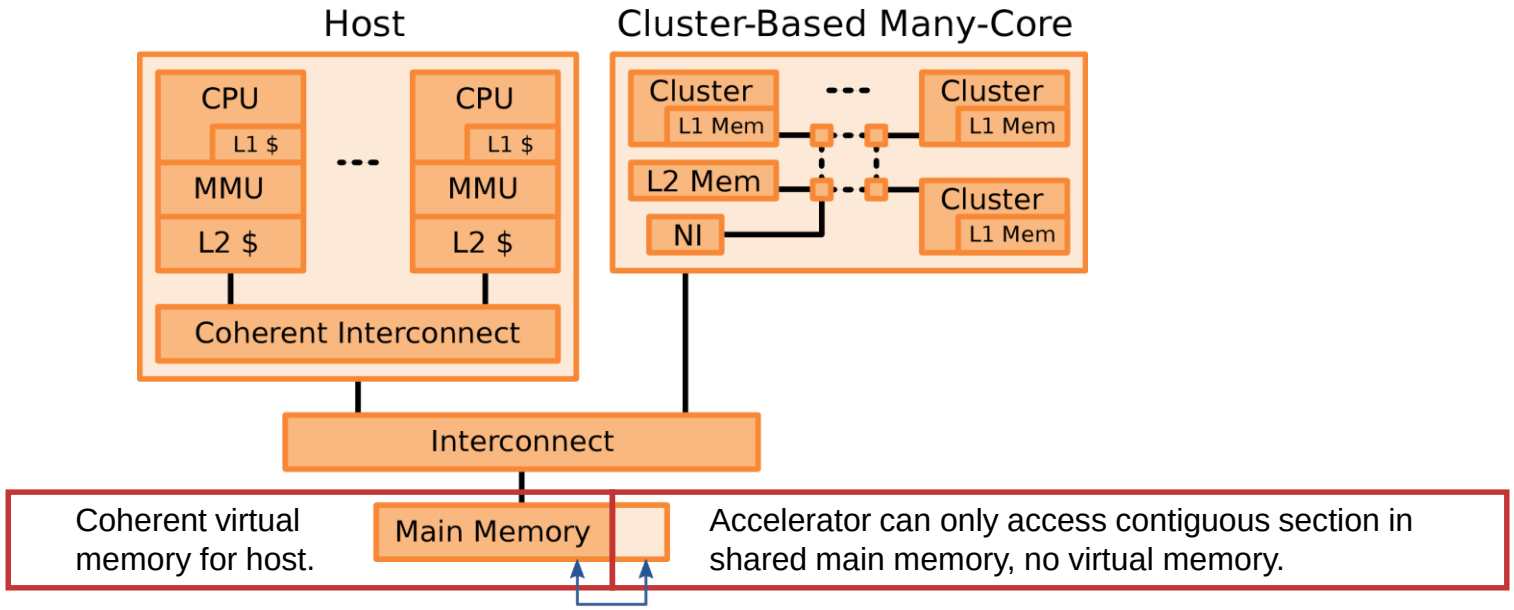
# Embedded Heterogenous SoCs

**copy-based approach**



# Embedded Heterogenous SoCs

**copy-based approach**




## Pros

- Do not require specific HW
- Cheap and low-power

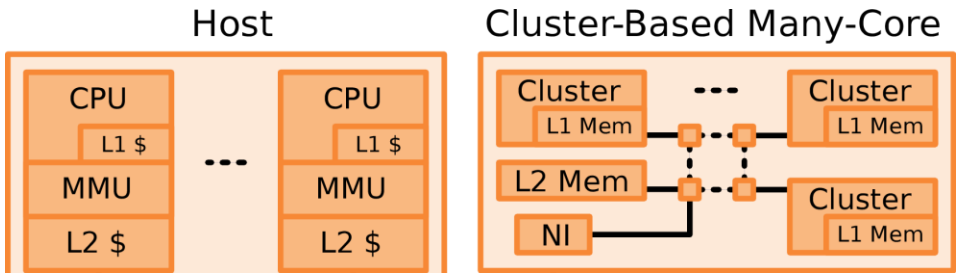
## Cons

- Overheads for copying data from/to the dedicated memory
- Complex data structures require ad-hoc transfer
- Performance issue on not-paged sections

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# Embedded Heterogenous SoCs

**copy-based approach**




| PageRank (10k vertices) | Execution Time | # Lines of Source Code |
|---|---|---|
| Host Implementation | 18.9 ms | 79 |
| Copy & Translate | 14.7 ms | 71 |
| Overhead | 78 % | 90 % |

## Pros

- Do not require specific HW
- Cheap and low-power

## Cons

- Overheads for copying data from/to the dedicated memory
- Complex data structures require ad-hoc transfer
- Performance issue on not-paged sections

# HERO lightweight shared virtual memory support

**Goals:**

- **Sharing of virtual address** pointers
- **Transparent** to application developer
- **Zero-copy offload**, performance predictability
- **Low complexity**, low area, low cost
- **Non-intrusive** to accelerator architecture

**Remapping Address Block (RAB):**

> Virtual-to-physical address translation

> Per-port private IOTLBs, shared configuration interface

## Mixed Hardware/Software Solution:

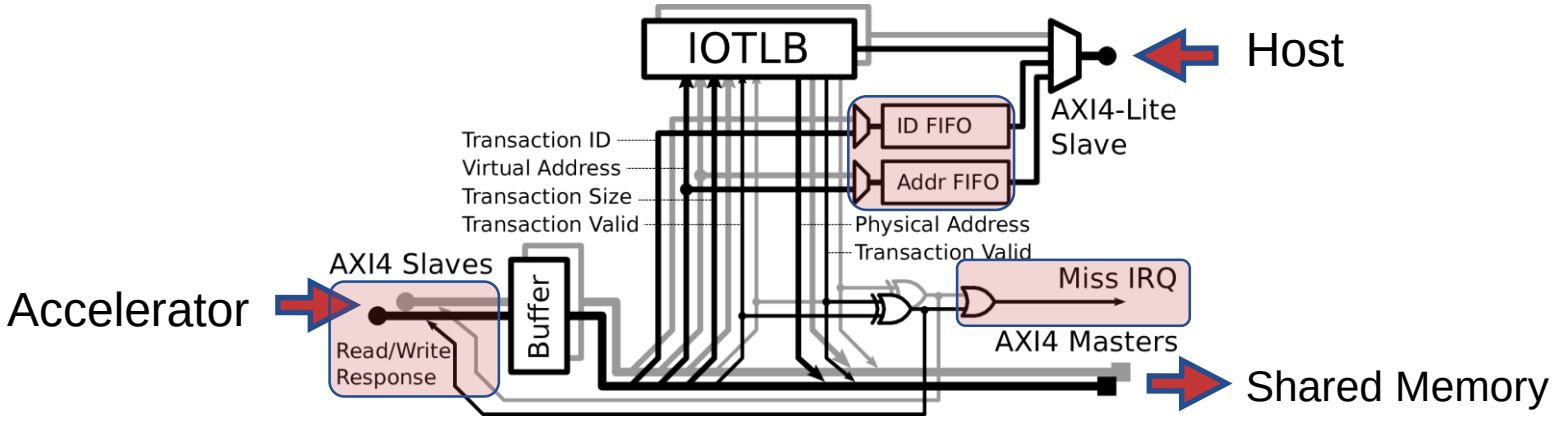
> Input/output translation lookaside buffer (IOTLB)

> Special-purpose *TRYX* Control register

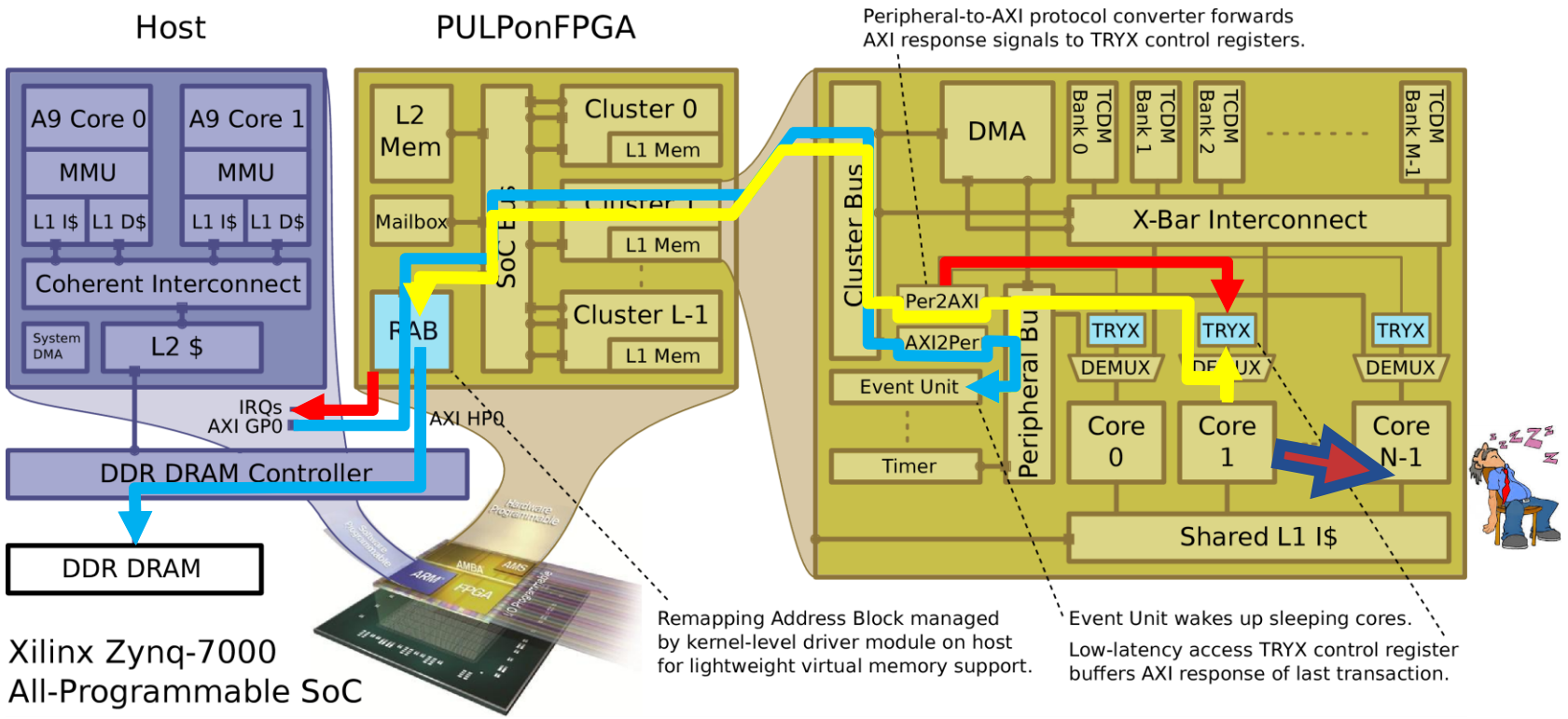
## Requires:

> Compiler extension to insert *tryread/trywrite* operation

> Kernel-level driver module



# HERO lightweight shared virtual memory support



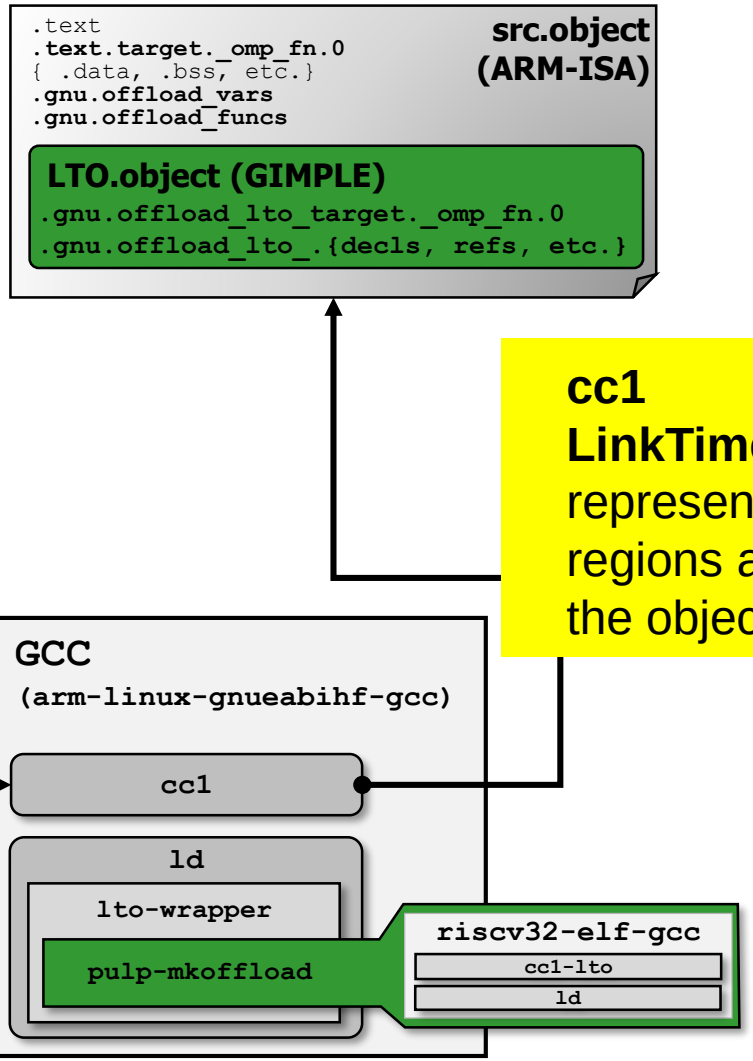

- **No hardware modifications** to the processing elements.
- **Portable RAB miss handling** routine on the host.
- Optimized for common case: **overhead of 8 cycles.**

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# Added HERO as target accelerator (1)

```
struct vertex {
    unsigned int vertex_id, n_successors;
    float pagerank, pagerank_next;
    vertex ** successors;
} * vertices;
...
#pragma omp target map(tofrom: vertices, n_vertices)
for ( i = 0; i < n_vertices; i++) {                    1
    vertices[i].pagerank = compute(... );
2   vertices[i].pagerank_next = compute_next(...);

    pr_sum += (vertices + i)->pagerank;

    if ((vertices+i)->n_successors == 0) {             3
        pr_sum_dangling += (vertices + i)->pagerank;
    }
}
```

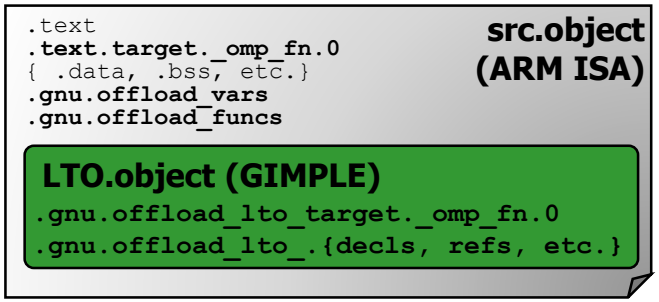
**ORIGINAL CODE**

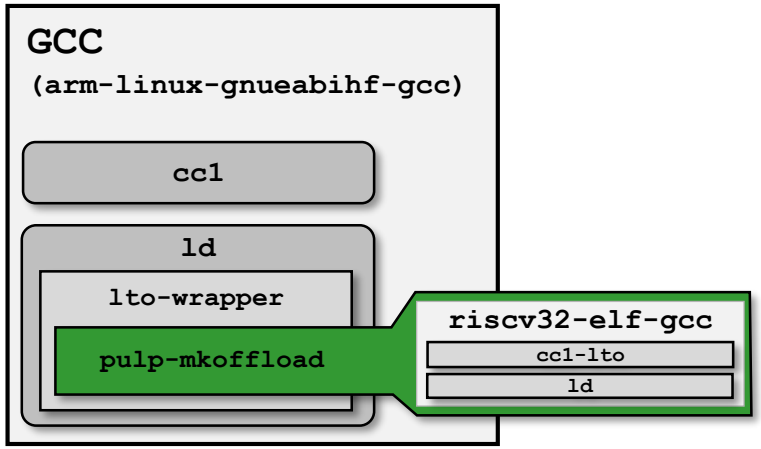



**cc1 LinkTimeOptimization** representation of target regions are appended to the object file

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# Added HERO as target accelerator (2)

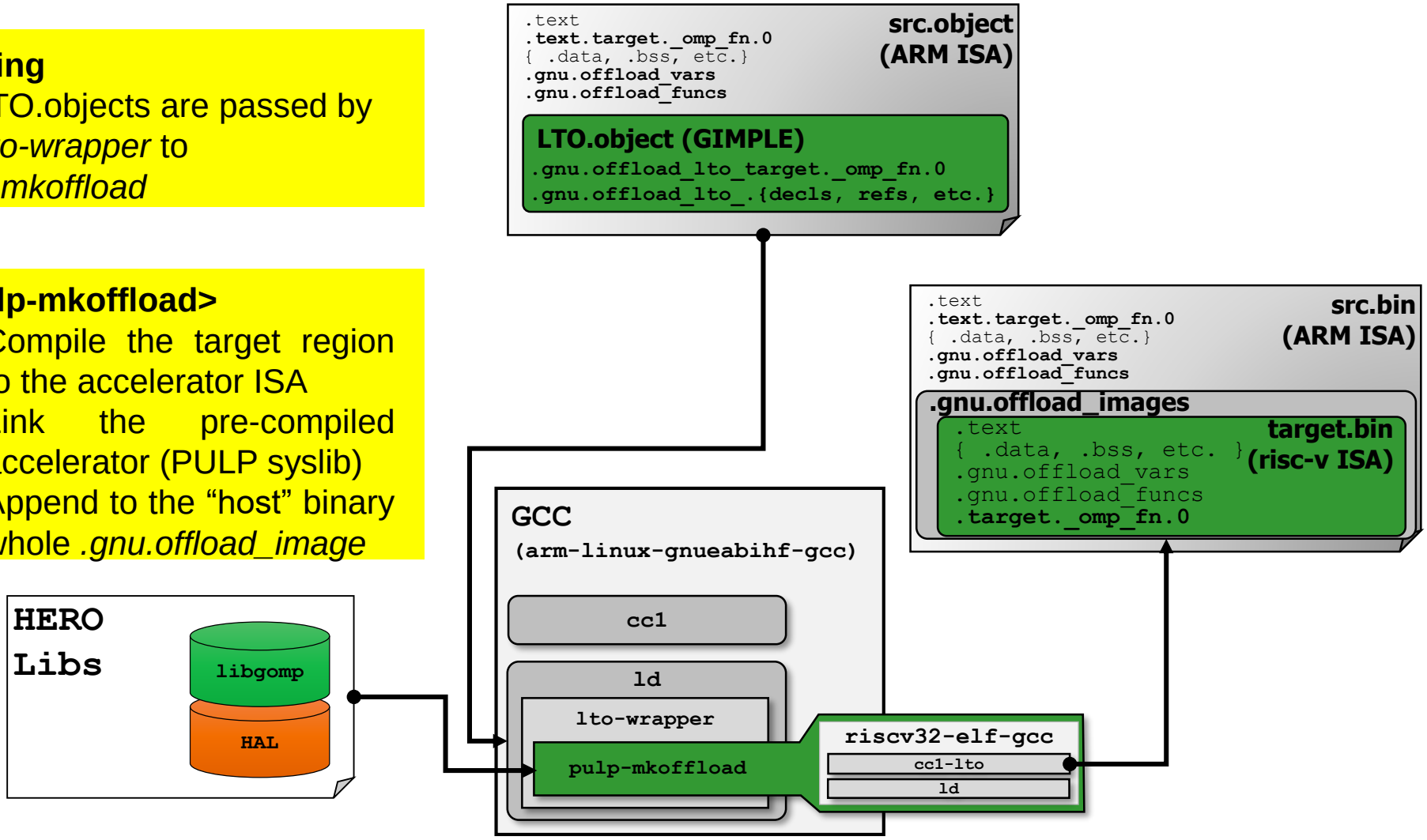


# Added HERO as target accelerator (2)

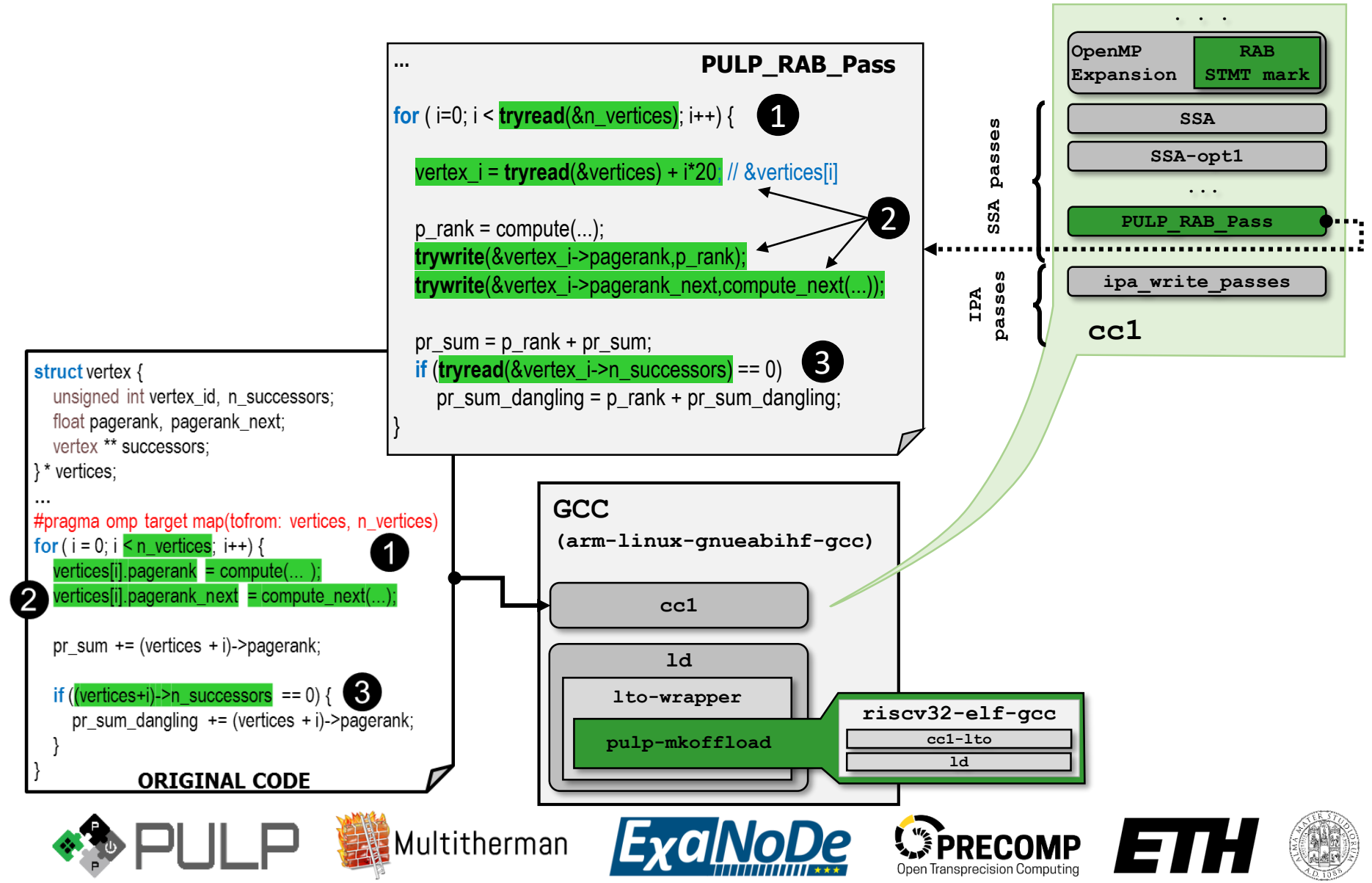
**Linking**
All LTO.objects are passed by the *lto-wrapper* to *pulp-mkoffload*

**<pulp-mkoffload>**
- Compile the target region to the accelerator ISA
- Link the pre-compiled accelerator (PULP syslib)
- Append to the "host" binary whole *.gnu.offload_image*



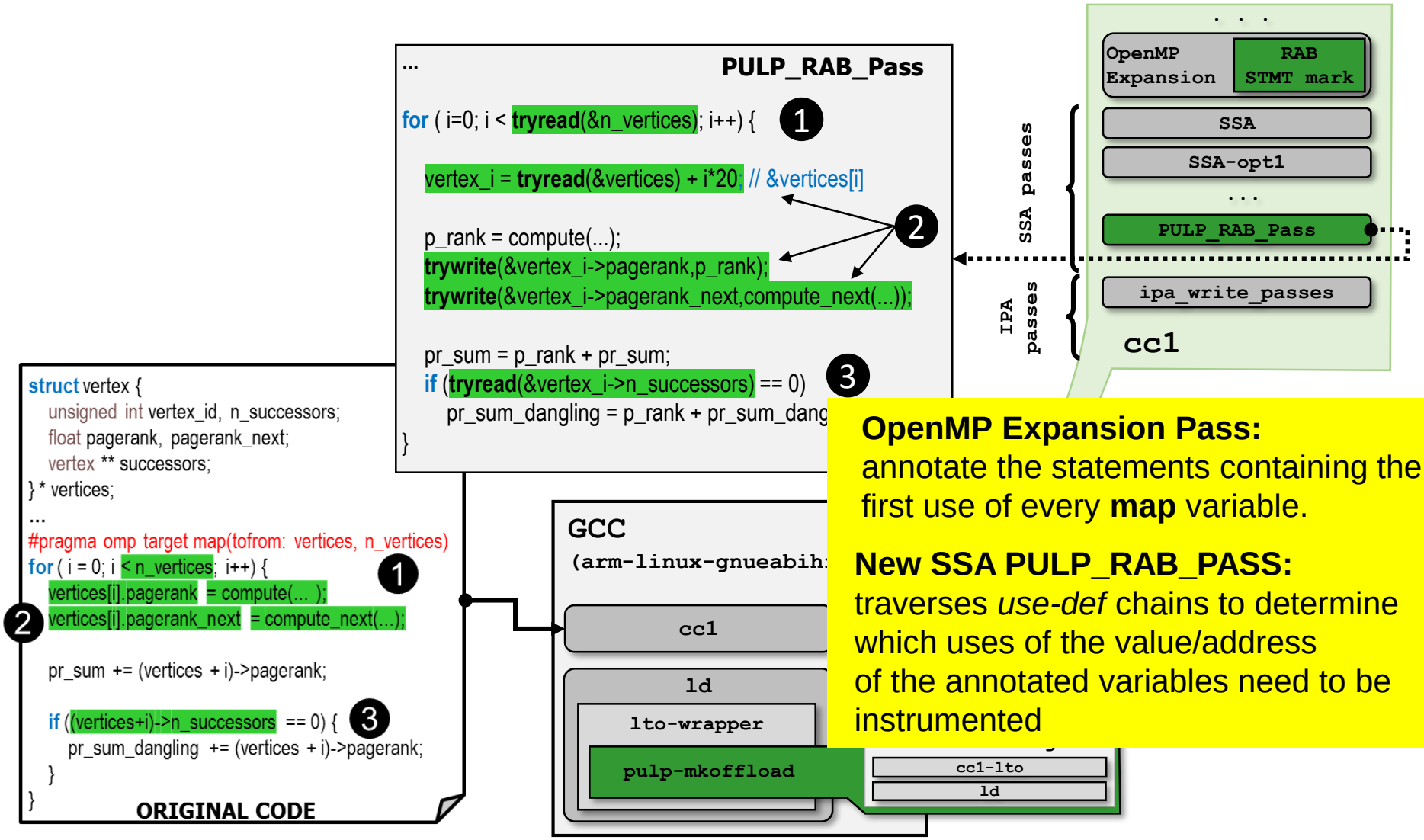

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# Compiler Shared Virtual Memory support




PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# Compiler Shared Virtual Memory support




**OpenMP Expansion Pass:**
annotate the statements containing the first use of every **map** variable.

**New SSA PULP_RAB_PASS:**
traverses *use-def* chains to determine which uses of the value/address of the annotated variables need to be instrumented

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

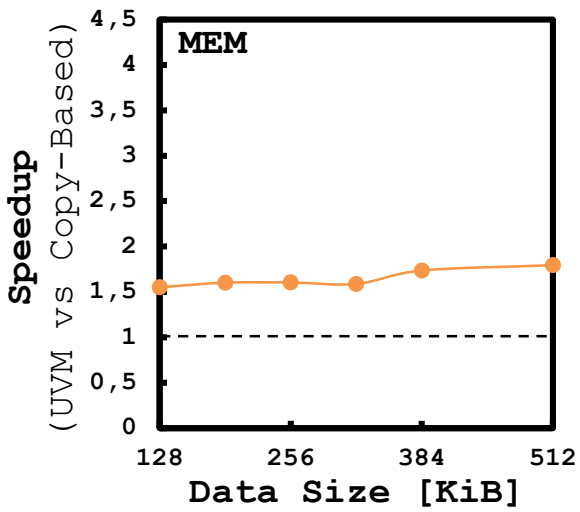
# Experimental setup

**Objective:** while Shared Virtual Memory greatest advantage is simplified programmability we want evaluate the advantage of Shared Virtual Memory on performance.

**Shared Virtual Memory Evaluation:**

- ***memcpy* (MEM*):*** representative example for heavily memory-bound, streaming applications with regular access pattern to shared memory.
- ***pointer chasing (PC):*** is representative of graph-processing applications with highly irregular access patterns, like Page-Rank, Breadth-First Search, clustering, Nearest Neighbor Search.
- ***random forest traversal (RFT):*** is representative of irregular applications for regression, classification problem solving, and pattern recognition

# Shared Virtual Memory Results (1)

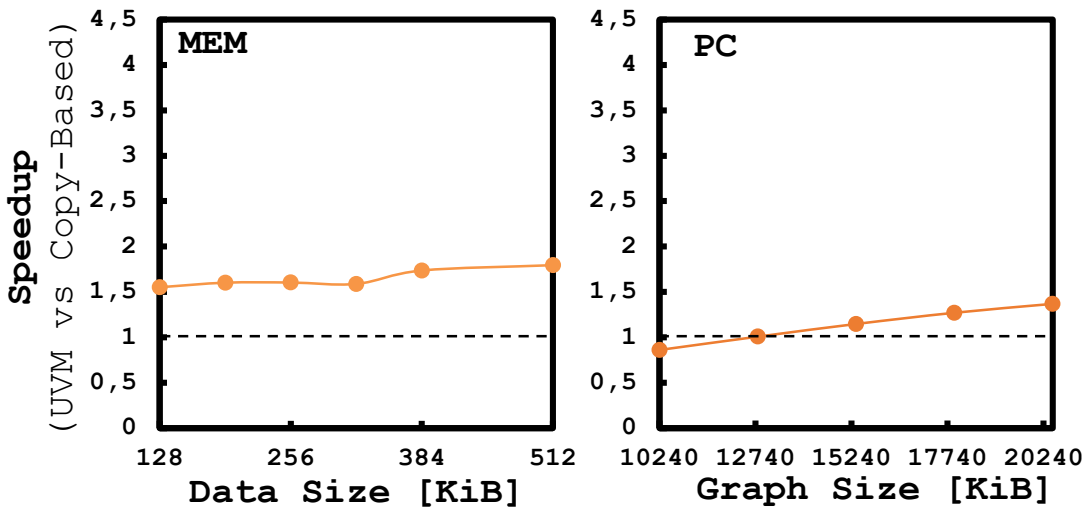



On regular applications SVM executes avg. 1.6× faster.

Capacity RAB misses when the data size exceeds the TLB capacity limits the speedup at 1.79×

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# Shared Virtual Memory Results (2)

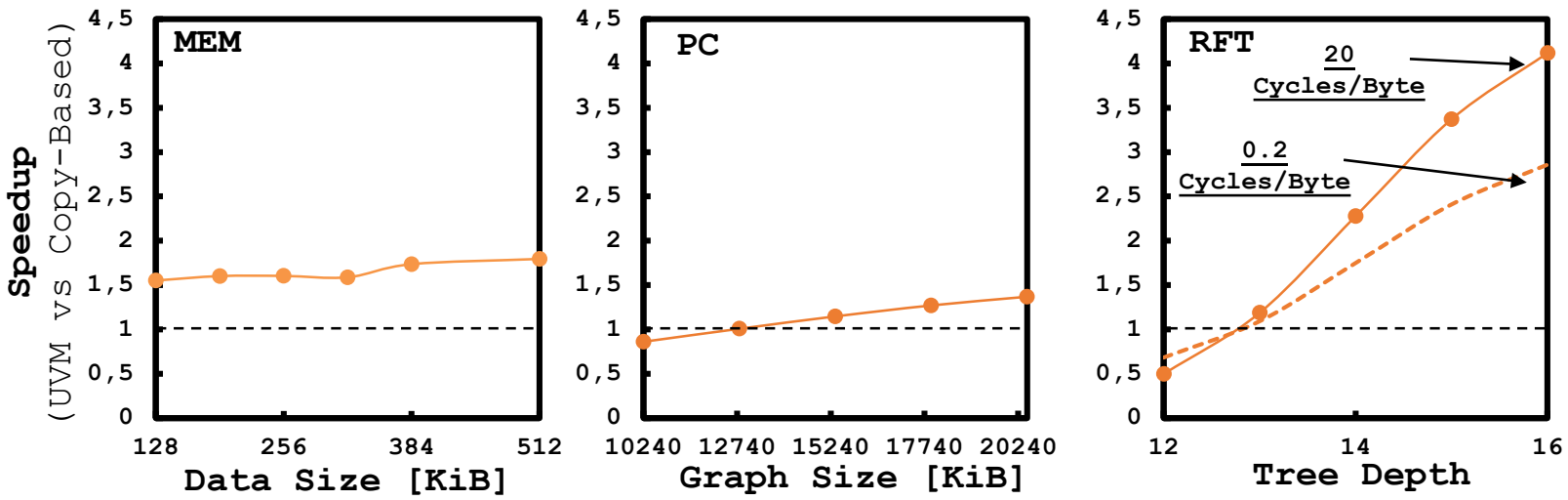



PC shows a slowly but steadily increasing speedup (up to 1.4× for the considered data sets)

Small graphs are penalized by the higher RAB handling costs compared to regular applications like MEM

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# Shared Virtual Memory Results (3)



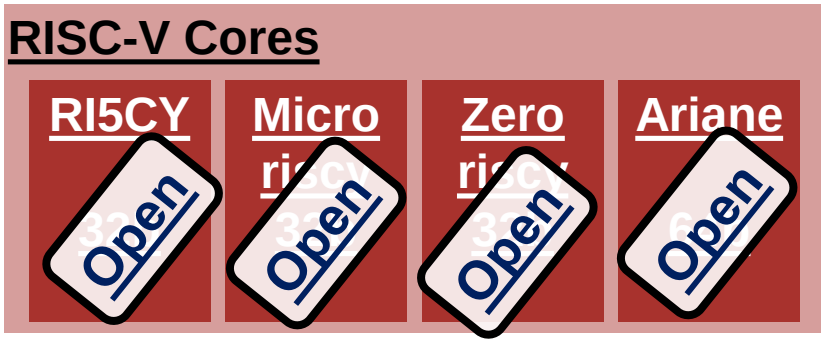

RFT reaches 4.11× and 2.85× speedup, for CCRs equal to 20 and 0.2, respectively

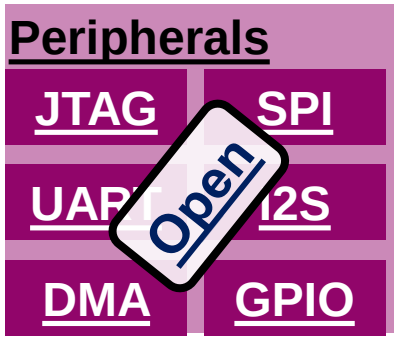
The higher speedups are due to the fact that in copy-based a lot of data is copied that is (potentially) never accessed.

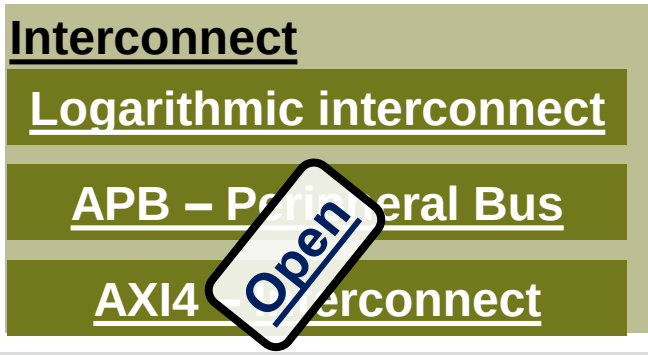
# Conclusion

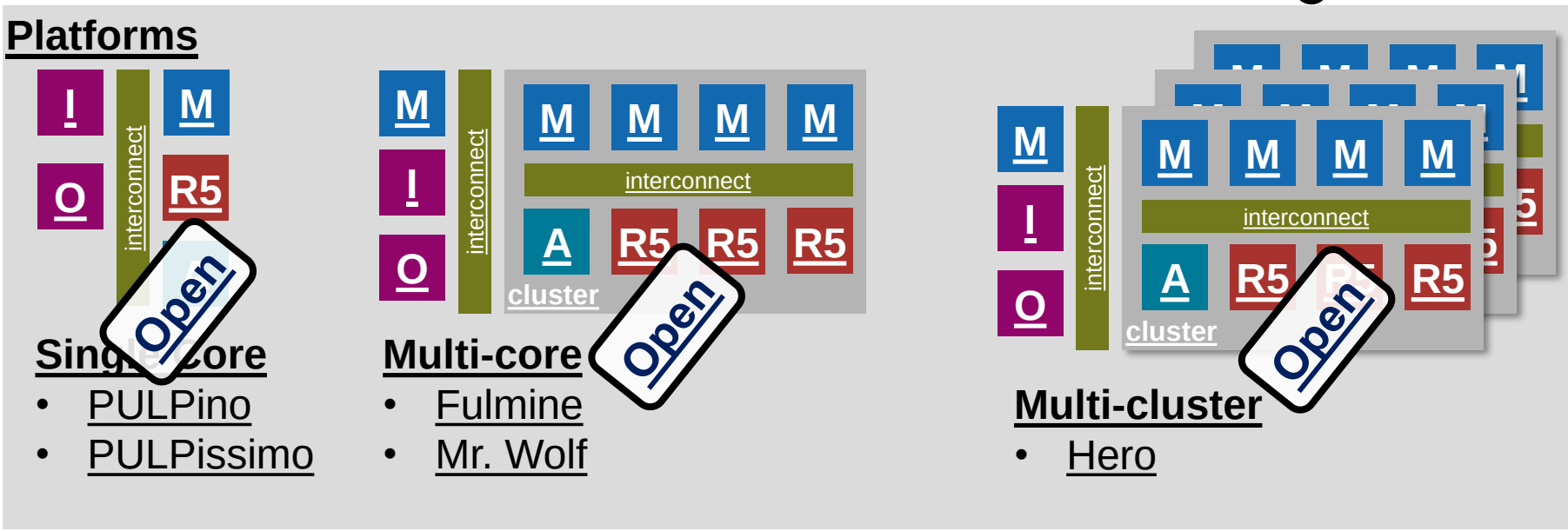
- We introduced HERO, a RISC-V based Open-Source Heterogeneous Research Platform
- We presented a **RTL-proven mixed HW/SW lightweight IOMMU** for low-power embedded many-core accelerator.
- We presented **a full implementation of OpenMP 4 on GCC** for PULP architecture.
- We **extended the toolchain at compiler and runtime level** to enable Shared Virtual Memory support achieved by a low-cost, low-area, IOTLB infrastructure.
- Shared Virtual Memory enables, with **smaller programming effort**, a **performance gain** compare **standard copy-based** offloading mechanisms.

# HERO as all PULP IPs is open-source!





pulp-platform.org/hero

# Thanks for your attention! Questions?

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# libgomp plugin determines how I/O variables are exchanged

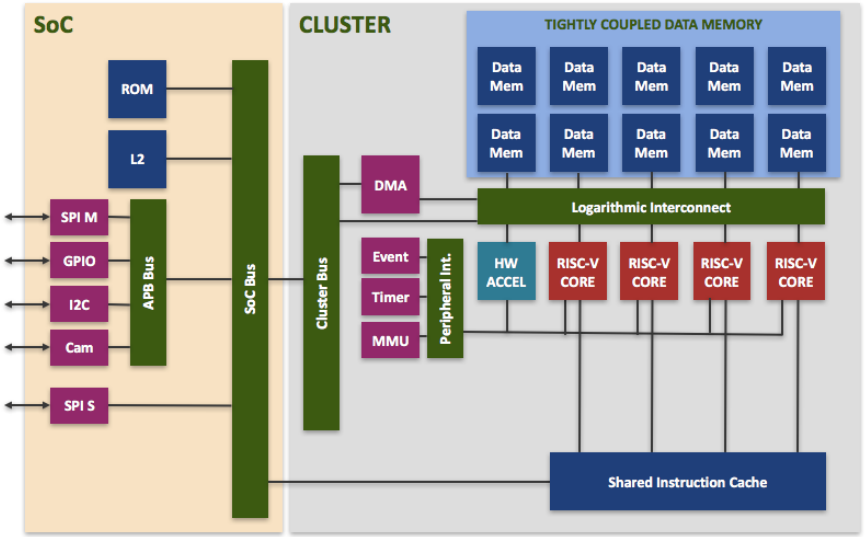
- **Copy-based shared memory**
  - Data is copied to a physically contigous uncached section of the main memory
  - Physical pointers are passed to PMCA
- **Shared virtual memory**
  - Enables zero-copy offloads, directly passing virtual pointers to PMCA
- libgomp plugin also implements essential OpenMP functionality
  - **parallel** (starting parallel execution)
  - **team** (definition of parallel thread teams)
  - **sections** (distributed, one-time execution worksharing)
  - **barrier** (syncrhonization barrier)
  - **critical** (single-threaded execution within a parallel region)

PULP
Parallel Ultra Low Power

## PULP - An Open Parallel Ultra-Low-Power Processing-Platform

This is a joint project between the Integrated Systems Laboratory (IIS) of ETH Zurich and the Energy-efficient Embedded Systems (EEES) group of UNIBO to develop an **open**, **scalable Hardware and Software** research platform with the goal to break the pJ/op barrier within a power envelope of a few *mW*.

The **PULP platform** is a multi-core platform achieving leading-edge energy-efficiency and featuring widely-tunable performance.




open-source
cluster-based
scalable
silicon-proven
RISC-V

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# Optimized RISC-V cores

**RISC-V Cores**

| RI5CY 32b | Micro riscy 32b | Zero riscy 32b | Ariane 64b |
|---|---|---|---|

Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# Hardware accelerators

**RISC-V Cores**

| RI5CY 32b | Micro riscy 32b | Zero riscy 32b | Ariane 64b |
|---|---|---|---|

**Accelerators**

| HWCE (convolution) | Neurostream (ML) | HWCrypt (crypto) | PULPO (1st order opt) |
|---|---|---|---|

# Peripherals and interconnect solutions

**RISC-V Cores**

- RI5CY 32b
- Micro riscy 32b
- Zero riscy 32b
- Ariane 64b

**Peripherals**

- JTAG
- SPI
- UART
- I2S
- DMA
- GPIO

**Interconnect**

- Logarithmic interconnect
- APB – Peripheral Bus
- AXI4 – Interconnect

**Accelerators**

- HWCE (convolution)
- Neurostream (ML)
- HWCrypt (crypto)
- PULPO (1st order opt)

# By combining these components we get PULP platforms

**RISC-V Cores**

- RI5CY 32b
- Micro riscy 32b
- Zero riscy 32b
- Ariane 64b

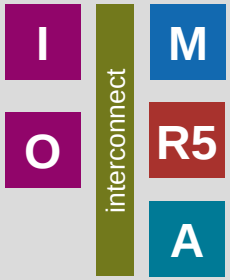
**Peripherals**

- JTAG
- SPI
- UART
- I2S
- DMA
- GPIO

**Interconnect**

- Logarithmic interconnect
- APB – Peripheral Bus
- AXI4 – Interconnect

**Platforms**




**Single Core**

- PULPino
- PULPissimo

**Accelerators**

- HWCE (convolution)
- Neurostream (ML)
- HWCrypt (crypto)
- PULPO (1st order opt)

# Our main research is on Near-Threshold Multi-Core Systems

**RISC-V Cores**

| RI5CY 32b | Micro riscy 32b | Zero riscy 32b | Ariane 64b |
|---|---|---|---|

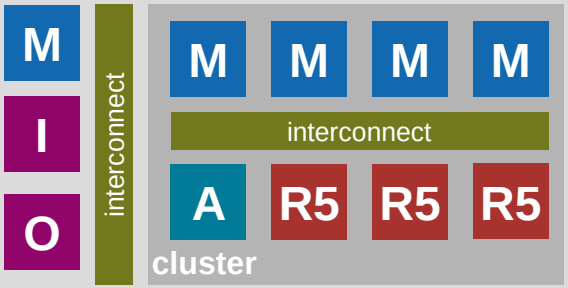
**Peripherals**

| | |
|---|---|
| JTAG | SPI |
| UART | I2S |
| DMA | GPIO |

**Interconnect**

- Logarithmic interconnect
- APB – Peripheral Bus
- AXI4 – Interconnect

**Platforms**


**Single Core**
- PULPino
- PULPissimo

**Multi-core**
- Fulmine
- Mr. Wolf

**Accelerators**

| HWCE (convolution) | Neurostream (ML) | HWCrypt (crypto) | PULPO (1st order opt) |
|---|---|---|---|

# Finally for HPC applications we have multi-cluster systems

**RISC-V Cores**

| RI5CY 32b | Micro riscy 32b | Zero riscy 32b | Ariane 64b |
|---|---|---|---|

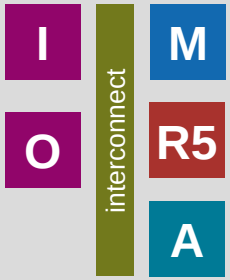
**Peripherals**

| | |
|---|---|
| JTAG | SPI |
| UART | I2S |
| DMA | GPIO |

**Interconnect**

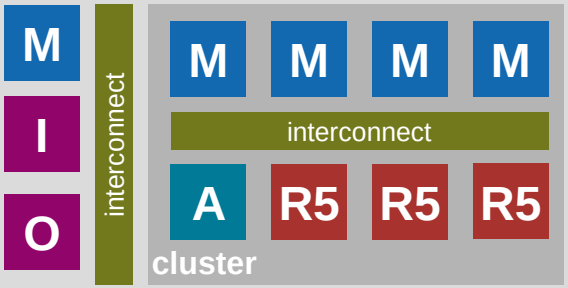
- Logarithmic interconnect
- APB – Peripheral Bus
- AXI4 – Interconnect

**Platforms**



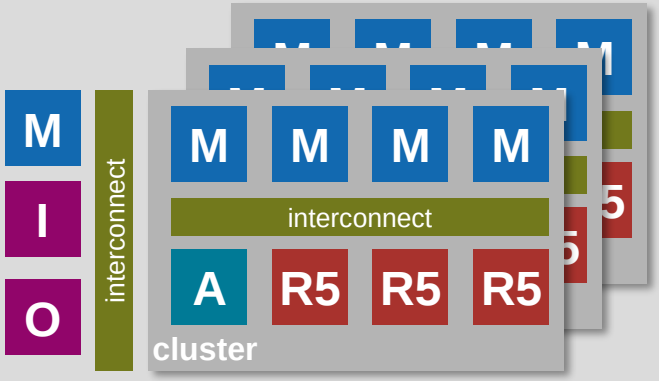

**Single Core**

- PULPino
- PULPissimo




**Multi-core**

- Fulmine
- Mr. Wolf




**Multi-cluster**

- Hero

IOT → HPC

**Accelerators**

| HWCE (convolution) | Neurostream (ML) | HWCrypt (crypto) | PULPO (1st order opt) |
|---|---|---|---|

# How-To Create an Heterogenous GCC Toolchain

- Target GCC Toolchain must be compiled with: "*--enable-as-accelerator-for=host-triplet*"
- Host GCC Toolchain must be compiled with: "*--enable-offload-targets=target1,target2,...*"

- The install locations for the offload compilers differ from those of a normal cross toolchain, by the following mapping:
  - bin/$target-gcc -> bin/$host-accel-$target-gcc
  - lib/gcc/$target/$ver/ -> lib/gcc/$host/$ver/accel/$target

- A target needs to provide a *mkoffload* tool if it wishes to be usable as an accelerator.
  - mkoffload will invoke the offload compiler in LTO mode to produce an offload binary from the host object files;
  - post-process this to produce a new object file that can be linked in with the host executable;
  - it can find the host compiler by examining the COLLECT_GCC environment variable, and it must take care to clear this and certain other environment variables when executing the offload compiler so as to not confuse it.

# Target RISCV32G Compiler Build

- Configure and Build RISCV32G Binutils

  ${HERO_ACCEL_SRC_DIR}/binutils/configure --target=${HERO_ACCEL_TARGET} --prefix=${HERO_ACCEL_GCC_INSTALL_DIR} --disable-werror

  make all ${HERO_PARALLEL_BUILD}

  make install

- Configure and Build RISCV32G Newlib

  ${HERO_ACCEL_SRC_DIR}/newlib/configure --prefix=${HERO_ACCEL_GCC_INSTALL_DIR} --target=${HERO_ACCEL_TARGET} --enable-newlib-io-long-double --enable-newlib-io-long-long --enable-newlib-io-c99-formats 'CFLAGS_FOR_TARGET=-Os -mcmodel=medlow' 'target_alias=riscv32-unknown-elf'

  make all ${HERO_PARALLEL_BUILD}

  make install

- Configure and Build First-Stage GCC Compiler

  ${HERO_ACCEL_GCC_SRC_DIR}/configure --target=${HERO_ACCEL_TARGET} --prefix=${HERO_ACCEL_GCC_INSTALL_DIR} --disable-shared --disable-threads --disable-tls --enable-languages=c,c++ --with-system-zlib --with-newlib --disable-libmudflap --disable-libssp --disable-libquadmath --disable-libgomp --disable-nls --enable-checking=yes --enable-multilib --with-abi=ilp32 --with-arch=rv32imc 'CFLAGS_FOR_TARGET=-Os -mcmodel=medlow' 'target_alias=riscv32-unknown-elf'

  make all-gcc ${HERO_PARALLEL_BUILD}

  make install-gcc

# Target RISCV32G Compiler Build - 2

- Configure and Build Whole GCC Compiler

  ${HERO_ACCEL_GCC_SRC_DIR}/configure --build=${HERO_BUILD_TARGET} --target=${HERO_ACCEL_TARGET} **--enable-as-accelerator-for=arm-linux-gnueabihf** --prefix=${HERO_ACCEL_GCC_INSTALL_DIR} --enable-languages=c,c++,lto --disable-shared --disable-threads --with-system-zlib --enable-tls --with-newlib --with-headers=${HERO_ACCEL_GCC_INSTALL_DIR}/${HERO_ACCEL_TARGET}/include --disable-libmudflap --disable-libssp --disable-libquadmath --disable-libgomp --disable-nls --enable-checking=yes --enable-multilib --with-abi=ilp32 --with-arch=rv32imc 'CFLAGS_FOR_TARGET=-Os -mcmodel=medlow' 'target_alias=riscv32-unknown-elf'

  make all ${HERO_PARALLEL_BUILD}

  make install

Take a look:

https://github.com/pulp-platform/hero-gcc-toolchain/blob/68bf73be8eeddc3ec99509c2a4dfe8f9a0c98576/hero_riscv32_toolchain_builder

# Target HERO mkoffload tool

- Located <gcc-sources>/*gcc/config/riscv/hero-mkoffload.c*

https://github.com/pulp-platform/pulp-riscv-gcc/blob/db8d182dc312b3a484cc6e6c1cb0260babd29867/gcc/config/riscv/hero-mkoffload.c

1. Find Target and Host Compilers
2. Prepare Target Image – Compile LTO to target binary representation
3. Generate Host offload Table Descriptor
4. Append target binary and Table Descriptor to the host ELF

# ARM Host Compiler Build

- Standard GCC linux-gnueabihf build
- Configuration is extended to add target compiler:

  ${HERO_HOST_GCC_SRC_DIR}/configure --prefix=${HERO_GCC_INSTALL_DIR} --target=${HERO_HOST_TARGET} **--enable-offload-targets=riscv32-unknown-elf=${HERO_GCC_INSTALL_DIR}** --with-arch=armv7-a ${HERO_HOST_FPU_CONFIG} --with-mode=thumb --enable-languages=c,c++,lto --disable-multilib --disable-nls --disable-werror --disable-sjlj-exceptions --with-sysroot=${SYSROOT}

# OpenMP target Expansion – 006t.omplower

helloworld.c.004t.gimple | helloworld.c.006t.omplower

```
;; Function helloworld (helloworld, funcdef_no=24, decl_uid=6918,
  cgraph_uid=24, symbol_order=29)

Introduced new external node (helloworld._omp_fn.0/31).
__attribute__((omp declare target))
helloworld ()
{
  {
    #pragma omp parallel [child fn: helloworld._omp_fn.0 (???)]
      {
        D.6928 = omp_get_thread_num ();
        D.6929 = omp_get_num_threads ();
        printf ("Hello World, I am thread %d of %d\n", D.6928, D.6929
          );
        #pragma omp return
      }
  }
}


;; Function main (main, funcdef_no=25, decl_uid=6922, cgraph_uid=25,
  symbol_order=30)

Introduced new external node (main._omp_fn.1/36).
main (int argc, char * * argv)
{
  int D.6936;

  {
    omp_set_default_device (1);
    {
      #pragma omp target num_teams(1) thread_limit(0) [child fn:
        main._omp_fn.1 (???)]
        {
          helloworld ();
          #pragma omp return
        }
    }
    helloworld ();
    D.6936 = 0;
    return D.6936;
  }
  D.6936 = 0;
  return D.6936;
}
```

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# OpenMP target Expansion – 012t.ompexp

helloworld.c.004t.gimple × helloworld.c.012t.ompexp ×

```
;; Function helloworld (helloworld, funcdef_no=24, decl_uid=6918,
   cgraph_uid=24, symbol_order=29)


OMP region tree

bb 2: gimple_omp_parallel
bb 3: GIMPLE_OMP_RETURN

Added new low gimple function helloworld._omp_fn.0 to callgraph
Introduced new external node (__builtin_omp_get_thread_num/32).
Introduced new external node (__builtin_omp_get_num_threads/33).
Introduced new external node (printf/34).
```

So far so good…

```
oworld._omp_fn.0, funcdef_no=26,
 order=31)

helloworld._omp_fn.0 (void * .omp_data_i)
{
  int D.6935;
  int D.6934;

  <bb 5> [0.00%]:

  <bb 3> [0.00%]:
  D.6934 = __builtin_omp_get_thread_num ();
  D.6935 = __builtin_omp_get_num_threads ();
  printf ("Hello World, I am thread %d of %d\n", D.6934, D.6935);
  return;

}


;; Function helloworld (helloworld, funcdef_no=24, decl_uid=6918,
   cgraph_uid=24, symbol_order=29)

Merging blocks 2 and 6
Merging blocks 2 and 4
__attribute__ ((omp declare target))
helloworld ()
{
  <bb 2> [0.00%]:
  __builtin_GOMP_parallel (helloworld._omp_fn.0, 0B, 0, 0);
  return;

}
```

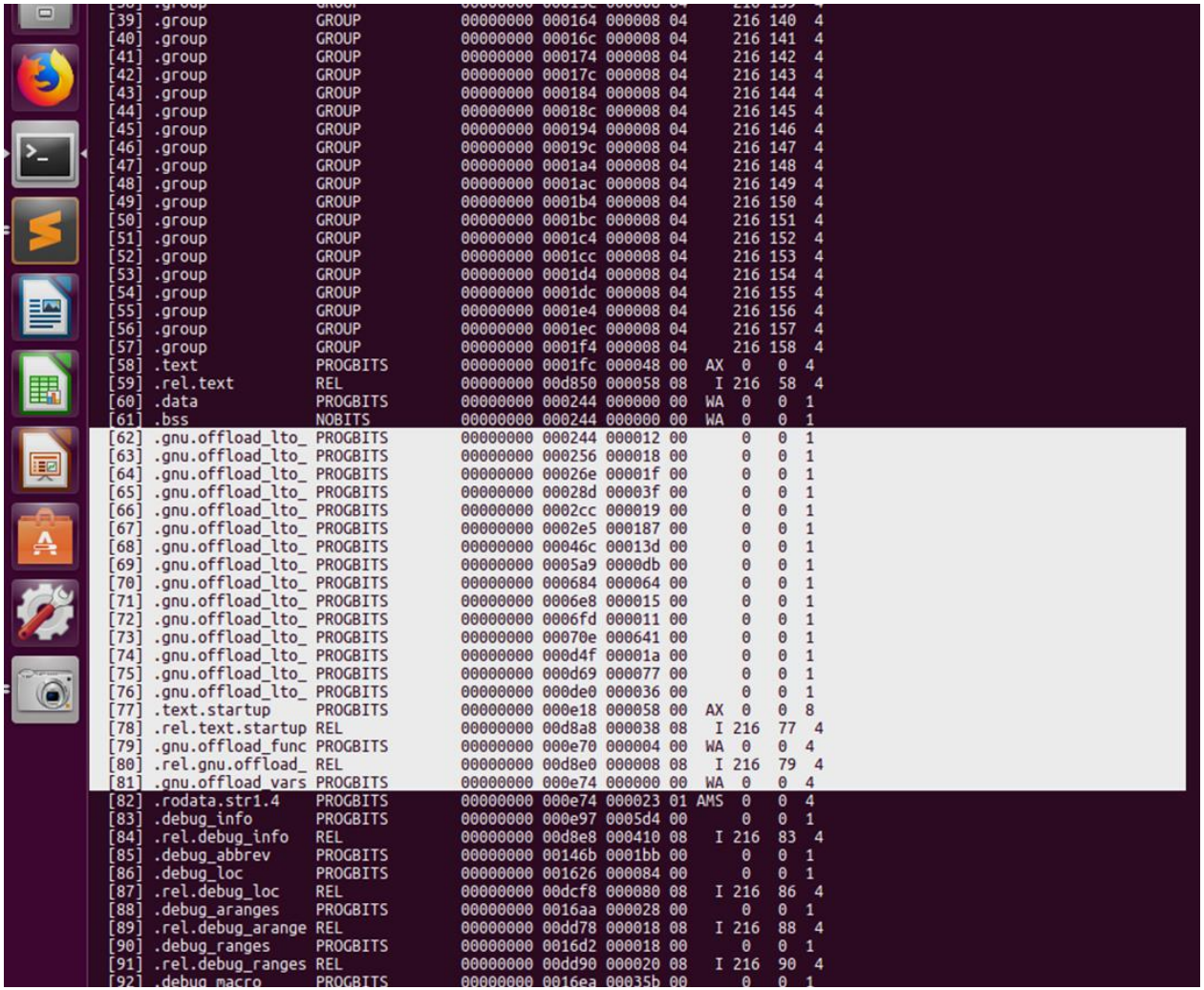
helloworld.c.004t.gimple × helloworld.c.012t.ompexp ●

```
;; Function main (main, funcdef_no=25, decl_uid=6922, cgraph_uid=25,
   symbol_order=30)


OMP region tree

bb 2: gimple_omp_target
bb 3: GIMPLE_OMP_RETURN

Added new low gimple function main._omp_fn.1 to callgraph

;; Function main._omp_fn.1 (main._omp_fn.1, funcdef_no=27, decl_uid=6938,
   cgraph_uid=31, symbol_order=36)

__attribute__ ((omp target entrypoint))
main._omp_fn.1 (void * .omp_data_i)
{
  <bb 5> [0.00%]:

  <bb 3> [0.00%]:
  helloworld ();
  return;

}

;;                                  l_uid=6922, cgraph_uid=25,

Me
Me
ma
{
  void * .omp_target_args.2[3];
  int D.6936;

  <bb 2> [0.00%]:
  omp_set_default_device (1);
  .omp_target_args.2[0] = 65792B;
  .omp_target_args.2[1] = 512B;
  .omp_target_args.2[2] = 0B;
  __builtin_GOMP_target_ext (-1, main._omp_fn.1, 0, 0B, 0B, 0B, 0, 0B, &.
    omp_target_args.2);
  helloworld ();
  D.6936 = 0;
  return D.6936;

}
```

GOMP_target_ext
Call to RTE for offloading

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH

# OpenMP target – Howto Manage multiple ISA? Readelf



```
[39] .group            GROUP           00000000 000164 000008 04     216 140  4
[40] .group            GROUP           00000000 00016c 000008 04     216 141  4
[41] .group            GROUP           00000000 000174 000008 04     216 142  4
[42] .group            GROUP           00000000 00017c 000008 04     216 143  4
[43] .group            GROUP           00000000 000184 000008 04     216 144  4
[44] .group            GROUP           00000000 00018c 000008 04     216 145  4
[45] .group            GROUP           00000000 000194 000008 04     216 146  4
[46] .group            GROUP           00000000 00019c 000008 04     216 147  4
[47] .group            GROUP           00000000 0001a4 000008 04     216 148  4
[48] .group            GROUP           00000000 0001ac 000008 04     216 149  4
[49] .group            GROUP           00000000 0001b4 000008 04     216 150  4
[50] .group            GROUP           00000000 0001bc 000008 04     216 151  4
[51] .group            GROUP           00000000 0001c4 000008 04     216 152  4
[52] .group            GROUP           00000000 0001cc 000008 04     216 153  4
[53] .group            GROUP           00000000 0001d4 000008 04     216 154  4
[54] .group            GROUP           00000000 0001dc 000008 04     216 155  4
[55] .group            GROUP           00000000 0001e4 000008 04     216 156  4
[56] .group            GROUP           00000000 0001ec 000008 04     216 157  4
[57] .group            GROUP           00000000 0001f4 000008 04     216 158  4
[58] .text             PROGBITS        00000000 0001fc 000048 00  AX  0   0  4
[59] .rel.text         REL             00000000 00d850 000058 08   I 216  58  4
[60] .data             PROGBITS        00000000 000244 000000 00  WA  0   0  1
[61] .bss              NOBITS          00000000 000244 000000 00  WA  0   0  1
[62] .gnu.offload_lto_ PROGBITS        00000000 000244 000012 00      0   0  1
[63] .gnu.offload_lto_ PROGBITS        00000000 000256 000018 00      0   0  1
[64] .gnu.offload_lto_ PROGBITS        00000000 00026e 00001f 00      0   0  1
[65] .gnu.offload_lto_ PROGBITS        00000000 00028d 00003f 00      0   0  1
[66] .gnu.offload_lto_ PROGBITS        00000000 0002cc 000019 00      0   0  1
[67] .gnu.offload_lto_ PROGBITS        00000000 0002e5 000187 00      0   0  1
[68] .gnu.offload_lto_ PROGBITS        00000000 00046c 00013d 00      0   0  1
[69] .gnu.offload_lto_ PROGBITS        00000000 0005a9 0000db 00      0   0  1
[70] .gnu.offload_lto_ PROGBITS        00000000 000684 000064 00      0   0  1
[71] .gnu.offload_lto_ PROGBITS        00000000 0006e8 000015 00      0   0  1
[72] .gnu.offload_lto_ PROGBITS        00000000 0006fd 000011 00      0   0  1
[73] .gnu.offload_lto_ PROGBITS        00000000 00070e 000641 00      0   0  1
[74] .gnu.offload_lto_ PROGBITS        00000000 000d4f 00001a 00      0   0  1
[75] .gnu.offload_lto_ PROGBITS        00000000 000d69 000077 00      0   0  1
[76] .gnu.offload_lto_ PROGBITS        00000000 000de0 000036 00      0   0  1
[77] .text.startup     PROGBITS        00000000 000e18 000058 00  AX  0   0  8
[78] .rel.text.startup REL             00000000 00d8a8 000038 08   I 216  77  4
[79] .gnu.offload_func PROGBITS        00000000 000e70 000004 00  WA  0   0  4
[80] .rel.gnu.offload_ REL             00000000 00d8e0 000008 08   I 216  79  4
[81] .gnu.offload_vars PROGBITS        00000000 000e74 000000 00  WA  0   0  4
[82] .rodata.str1.4    PROGBITS        00000000 000e74 000023 01 AMS  0   0  4
[83] .debug_info       PROGBITS        00000000 000e97 0005d4 00      0   0  1
[84] .rel.debug_info   REL             00000000 00d8e8 000410 08   I 216  83  4
[85] .debug_abbrev     PROGBITS        00000000 00146b 0001bb 00      0   0  1
[86] .debug_loc        PROGBITS        00000000 001626 000084 00      0   0  1
[87] .rel.debug_loc    REL             00000000 00dcf8 000080 08   I 216  86  4
[88] .debug_aranges    PROGBITS        00000000 0016aa 000028 00      0   0  1
[89] .rel.debug_arange REL             00000000 00dd78 000018 08   I 216  88  4
[90] .debug_ranges     PROGBITS        00000000 0016d2 000018 00      0   0  1
[91] .rel.debug_ranges REL             00000000 00dd90 000020 08   I 216  90  4
[92] .debug_macro      PROGBITS        00000000 0016ea 00035b 00      0   0  1
```

PULP Multitherman ExaNoDe PRECOMP Open Transprecision Computing ETH