Compilers Challenges for Heterogeneous Architectures

(or compilers challenges explained to hardware architects)
FETCH 2022, Les Houches

Henri-Pierre CHARLES

CEA DSCIN department / Grenoble

Vendredi 17 Juin 2022

Intro : Presentation Bridge

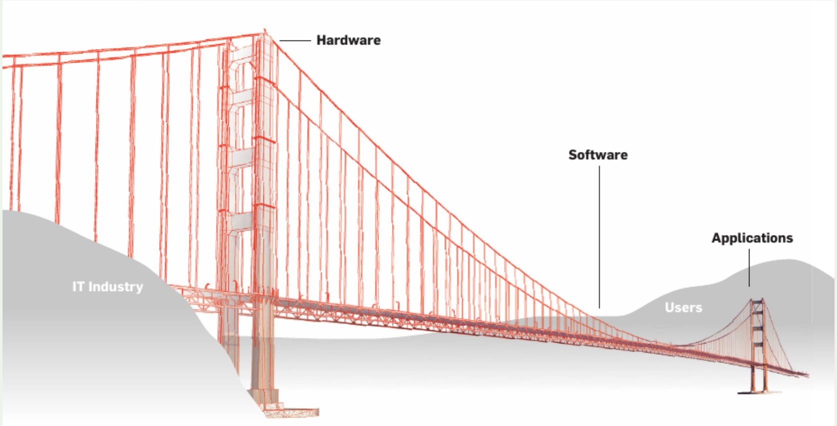


Figure 2. Bridge analogy connecting users to a parallel IT industry, inspired by the view of the Golden Gate Bridge from Berkeley, CA.

K. Asanovic et al., "A View of the Parallel Computing Landscape", Commun. ACM, vol. 52, nro 10, p. 56–67, oct. 2009.

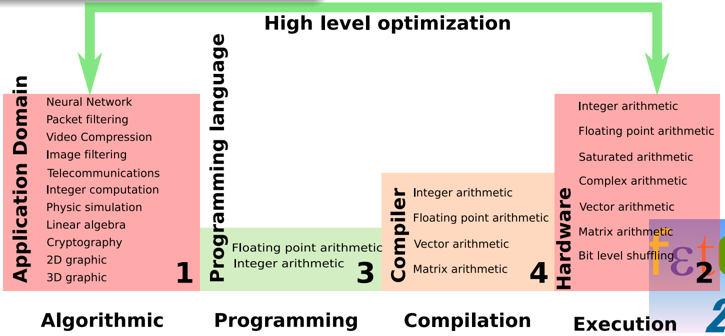
Introduction : What's the problem ?

Programming language

- Are invented and **standardized** between 70 and 90'
- Based on logic, integer and floating point arithmetic.
- Compiler use arithmetic rules for optimization

Computer architecture

- Use real world datasets : bitmap images, vector images, datagram, ...
- Hardware development and Low level programming research domains are deconnected



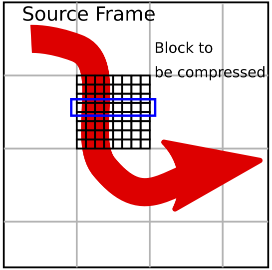
Introduction : Problem Example Image Compression

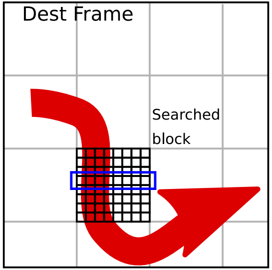
ARM specialized instruction

- `usad8 rd, rs1, rs2` (Page 4482 / 7476)
- Used in all video compression tools (VLC, FFMPEG)

Image compression basic : block comparison

- Res $= \sum_{i=0}^{8} |in0[i] - in1[i]|$
- No compiler support, should use assembly level programming
- (See Videolan VLC project)
- Software support :
  - Assembly level function : no compiler support
  - Programmer tiling model : no compiler support





Models : C Language for Architecture

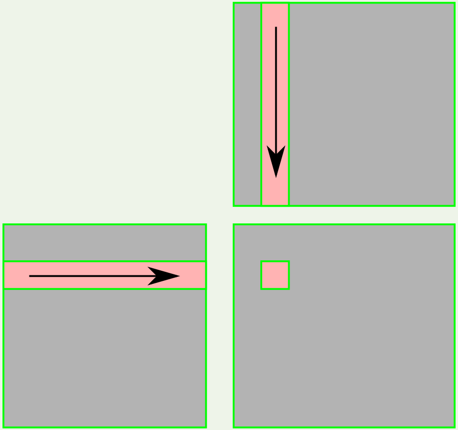
Matrix multiply (sketch)

```
for (int l = 0; l < SIZE; l++)
  for (int c = 0; c < SIZE; c++)
    for (int k = 0; k < SIZE; k++)
      R[l][c] += A[l][k] * B[k][c];
```

"Real world"

```
for (c= 0; c<NCOL; c+=cacheLineSize)
 for (l= 0; l<NLINE; l+=halfCacheLine)
  for (c2= 0; c2<NCOL; c2+=halfCacheLine)
    for (lk= 0; lk<halfCacheLine; lk++)
     for (c2k= 0; c2k<halfCacheLine; c2k++)
      for (ck= 0; ck<cacheLineSize; ck++)
        res[l+lk][c2+c2k]+=     a[l+lk] [c+ck]* b[c2+c2k][c+ck];
```

Learn to program = learn to serialize / schedule on defined hardware !



Other "interesting examples"

Introduction : C Data Types

Basic Data representation

Integers Binary-coded_decimal, Two's_complement (C)

Characters Baudot_code (1901), ASCII (1970) (C), Latin 1 1998, UTF-8 1992,

Floating point Intel, Ibm, IEEE (C), Unum, Posit, VRP, Stochastic

Others : Pixels (RGV, YUV, CMJN, ...), IP addresses, IA Objects (Cats, dogs, ...)

Constructed Data representation

Integer array index

Characters Strings, Text

Chaînes de caractères Chaînes, Texte

Floating point matrix, vector

Pixels Images (array of struct, struct of array), Sprites,

TCP packet ../.. (look at BPF : compilation for packet filtering)

Type Based Compiler Optimisations

Integer Algebra rules : commutativité, distributivité

Character None

Floating point None, unless using `-fast-math` which broke algorithms based on numerical stability

Pixels None

IP addresses None (BPF DSL specialized compiler)

IA objects Human slaves

Constructed Data Representation

- Compiler has not clue of the semantic of the constructed datatype.
- "Leave the axe to the programmer" !
- Please give complex data type semantic to a compiler

Scientific Evolution : Compilation Research Domains

Compilation Topics Map

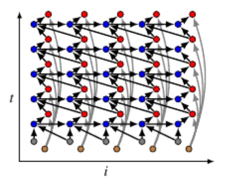
- Find code structure: Extract parallelism : polyhedral approach
- Assertion on legacy code: correctness proof, hard realtime, model checking
- Security: HW attach counter mesure, obfuscation
- Tools for scalability: Systems and library for big parallel machines
- Reproductibility: Statistics tools and reproductible research
- Ad hoc code optimization: Application driven code optimization
- Legacy compiler optimization: follow the HW evolution
- New code generation paradigm: JIT, Dynamic code generation

https://top500.org

LAPACK June 2002 #1 : HPE Cray 8,730,112 cores, Linpack perf 1,102.00 PFlop/s, 65% of peak performance (usualy better)

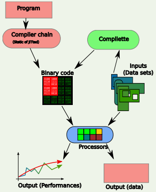
HPCG Fugaku 7630848 cores, 16,004 TFlops 2.98 % of peak performance !

Polyhedral model



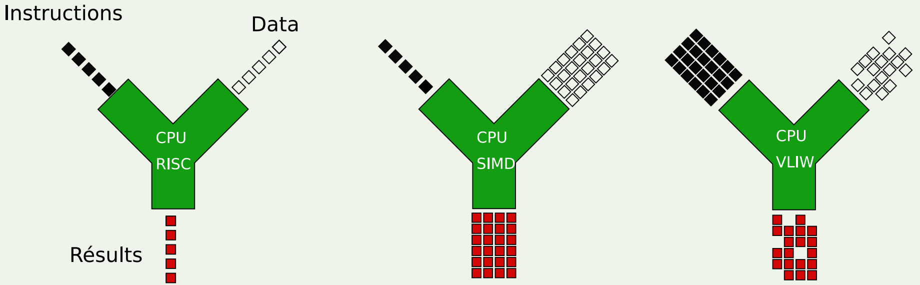
(a) CDAG for $T = 4, N = 7$

Algorithmic accelerator



Processor Words and Instructions Size

How to deal with low level parallelism



- Scalar RISC CPU : “simple” for compiler
- SIMD CPU aka Vector instructions : good for dense computation
- VLIW CPU : good for complex workload

CPU Programming Model

What's inside the programmer head ?

- Not only the ISA + addressing modes
- "Some" processors notions :
  - Registers
  - UALs / vectors width / # parallel UAL
  - Pipeline
  - Branch penalty
  - Caches L1 $D $I / Caches L2 / Caches L3
  - Interprocessors communication bandwith versus Latency
  - ...

MC 68000 "Programming model"

Freescale Semiconductor, Inc.

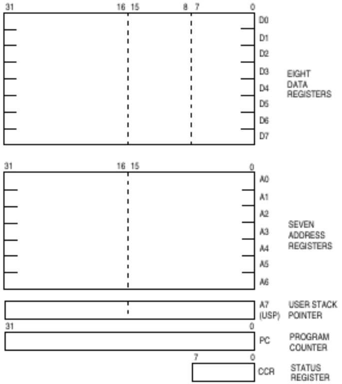


Figure 2-1. User Programmer's Model (MC68000/MC68HC000/MC68008/MC68010)

HWParallelismLevel uArch

What every programmer should know about performance

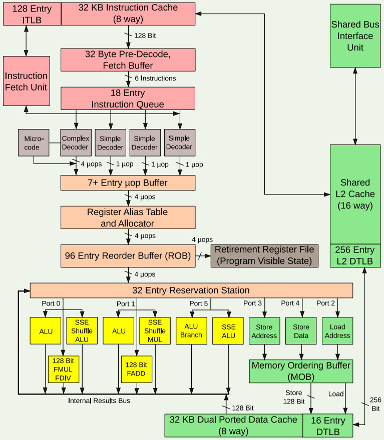
- Hidden micro architecture
- Memory hierarchy

Intel Example

- “700” simple high level instructions
- 17000+ instructions variants
- RISC internal uInstructions

Intel Micro architecture

Intel Core2 micro arch



Intel Core 2 Architecture

Domain Specific Language : HybroLang

By The Way :

Do we need a new programming language for a new progamming model ?
YES !
[Hennessy-Patterson "A New Golden Age for Computer Architecture"]

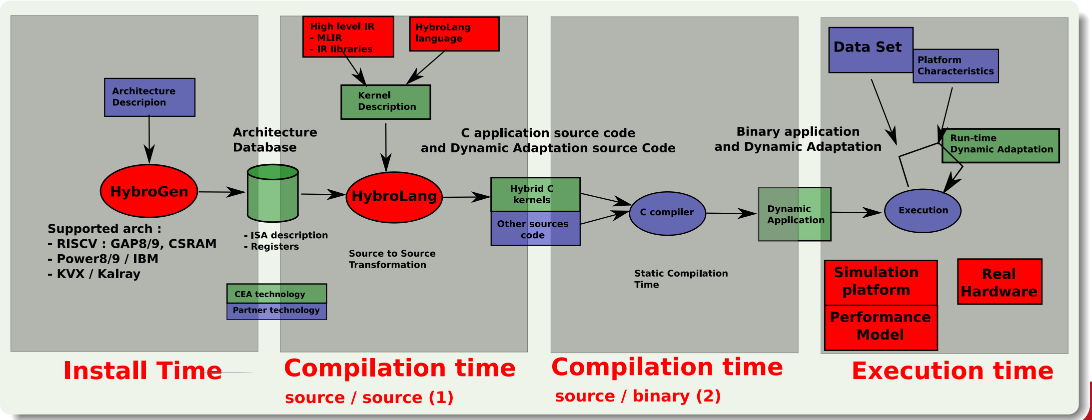
Why a new programming language ?

- Want to make data dependent code generation, on the fly.
- Do not want to find vectorization / parallelization : use it !
- Want to **implement** "Inverted Von Neumann Model" (slides on IMC)
- Want to use specific arithmetic : integer, saturated, arithmetic, ... IP@, stochastic, geometric shapes, ...

HybroLang features

- Similar to C syntax
- Variables = Hardware elements (register, memory line)
- Arithmetic Operators = existing processor UAL
- Run-time delayed code generation

HybroGen: General View



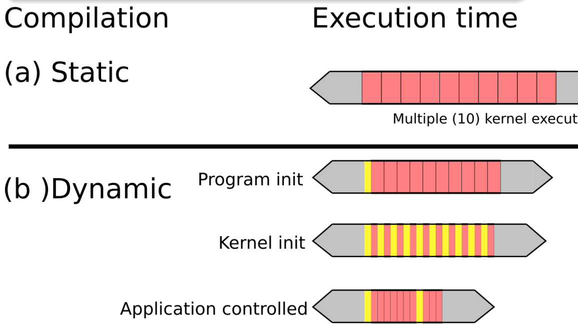
HybroGen : Objectives

Already done

- On the fly code generation for heterogeneous architectures
- Transprecision support : on the fly code generation for precision adaptation. Working demonstration on Newton algorithm : Power / RISCV / Kalray
- Support for In Memory Computing (next slides)
- Target processor modeling (QEMU plugin)

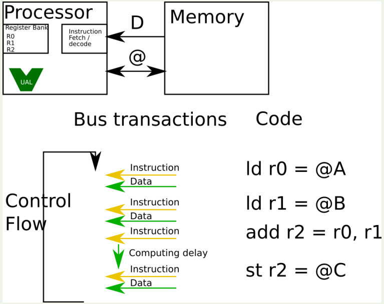
Application domains

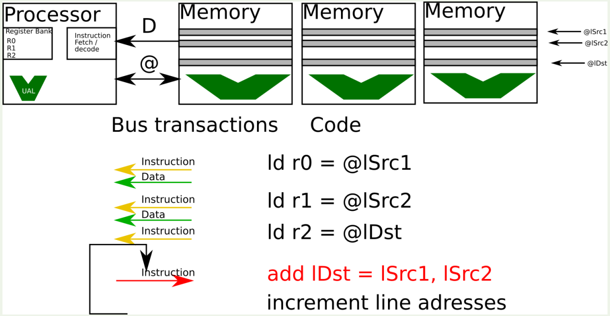
- Stochastic number support
- Packet filtering : Datatype ipv4, ipv6 addresses
- Transprecision algorithms : usage on mathematical iterative methods
- Stencil processing



Inverted Von Neumann Programming Model

Choosen Programming model





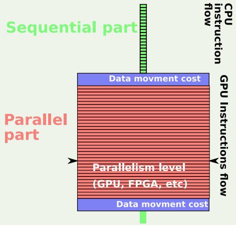
Why ?

- Allows scalability :
  - Any vector size
  - Any tile number
  - Any system configuration : near or far IMC
- Works with any processor

Von Neuman broked, what about Amhdal Law's ?

Ahmdal law's: "Speedup is limited by the sequential part"

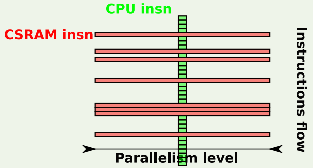
Classical approach



Programmer approach

- Has to maximize parallel part
- Deal with data "choregraphy" between CPU and GPU.

CSRAM approach



Programmer approach

- Ease to interlace scalar instruction and IMPACT instructions
- Do not move data
- No needed syncrhonization

Programming Model : Image Diff

Mini code Example : HybroLang code example

```
int 8 16 ImageDiff(int[] 8 16 a,
                   int[] 8 16 b,
                   int[] 8 16 res, int 31 1 len)
{
    int 31 1 i;                             /* Scalar variable */
    for(i = 0; i < len; i = i + 1)  /* CPU Control */
    {
        res[i] = a[i] - b[i];              /* IMC instructions */
    }
}
```

Compiler support

- Dynamic interleaving
- Instruction generator generator notion

Compiler support for System Architecture

Already investigated compiler support

- Ongoing Kevin Mambu PhD. Hypothesis :
  - 1 IMC tile + 1 IOT node (mono CPU)
  - 1 IMC tile + 1 CPU (with caches)
  - PhD to be presented in oct. 22
- Ongoing work : backend support for TensorFlow

Already investigated HW

- Multi tile IMC reconfigurable + 1 IOT node (mono CPU) Roman Gauchi PhD (done)

To be investigated

Scientific questions : do we need a new programming model ?

- CSRAM in DRAM scenario (PhD Valentin Egloff)
- CSRAM in image capture device (TbD)
- MultiTile CSRAM in MPSoC (TbD)
- Software support for data layout (TbD)
- ...

fεtCh 2022 cea

Targeted Applications & Domain

Already Investigated

- Image processing : many OpenVX Vision Function already optimized (8 / 43)
- Integer BLAS (Basic Linear Algebra Subprograms)
- In Memory Cryptography

To be Done

- Hash functions : Cryptography, Cryptocurrency
- Network applications : packet filtering
- DNA search / align
- Vectorisation for hard real time

Research Model Organization

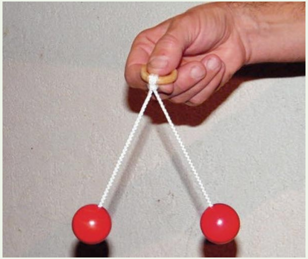
Actual Organization

- Research in memory design
  - Build memory chips (integrate High Level evaluation, specialized instructions)
  - Characterize behavior
- Research in compilation
  - Build simulation model (integrate characterization)
  - Build Software Tools
  - Evaluation on High Level Applications
- Multiples targets :
  - Support current & old chips
  - Invent / Test / support futures features
  - Evaluate futures features, using previous characterization
  - Generate test codes

Lessons learned

- Do not work on “one project”
- Connect & Interact, ... frequently
- Sort of agile Research Model

Freedom to move but interact frequently



Conclusion and Valorisation

Publication / patents

- Papers (SW / HW),
- PhD (1 defended, 3 ongoing)
- Patents : 10+
- CEA "Fait Marquant" : HW / SW
  - 2019 : First In-Memory Computing tile
  - 2021 : Compiler for «computing continuum», application for «In Memory Computing»
  - 2022 : Automatic test code generation from ISA description
  - 2022 : Second circuit tape out June 2022 : RISC-V + IMC
- External requests : Workshops, ANR review, Articles review

Results

- HybroGen Compilation Chain
  - OpenSource CECILL-B licence : https://github.com/CEA-LIST
  - To appear 15 days
- IMC Instruction Set : 4 release
- IMC Chips : 2 chips
- IMC Characterization

Important message

- Major players are vertical industries
- Application (& software) drive the market

Conclusion : Embauches au CEA Grenoble

CDD

- Poste de CDD “Innovation en compilation” ../.. travailler sur un compilateur qui permet de mettre en oeuvre des solutions innovantes de compilation : compilation dynamique, compilation pour machine non von-neumann, instruction spécialisées

Thèse

- “Simulation de systèmes numériques contenant un accélérateur quantique” ../.. constituer un simulateur de haut niveau, permettant de tester ces différentes hypothèses de micro architecture.
- “Micro-compilation pour réseaux de neurones ternaires sur une architecture de calcul proche mémoire” ../.. mettre au point un modèle de programmation innovant spécifique à ce type d’architecture

mailto:Henri-Pierre.CHARLES@cea.fr

fεtCh 2022