A Code Selection Method for SIMD Processors with PACK Instructions

Hiroaki Tanaka, Shinsuke Kobayashi,
Yoshinori Takeuchi, Keishi Sakanushi, Masaharu Imai
Integrated System Design Laboratory
Osaka University

Outline

- Introduction
- Previous work (Leupers' method)
  - Code selection with SIMD instructions
  - Limitation
- Proposed method
  - Code selection with SIMD and PACK instructions
- Experimental results
- Conclusion

Background

- Features of multimedia application
  - Instruction level parallelism
  - Handling shorter data types than word length
- Multimedia application oriented processors
  - SIMD Instructions, PACK Instructions
- Requirements to compilers
  - High quality of code
  - To take advantage of SIMD and PACK instructions

Proposal

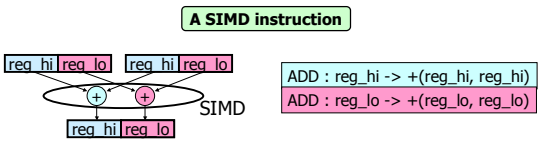
- Difficulties to handle SIMD and PACK
  - Detection of parallelism
  - Management of data location in registers
- Our proposed method
  - Code selection method with SIMD and PACK considering data parallelism and data location
- Key idea
  - Extension of DFT to represent data transfer between registers

Outline

- Introduction
- Previous work (Leupers' method)
  - Code selection with SIMD instructions
  - Limitation
- Proposed method
  - Code selection with SIMD and PACK instructions
- Experimental results
- Conclusion

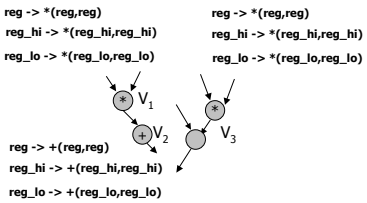
Induction Rules for SIMD Instructions [Leupers]

- Using DFT and tree grammar
- Introducing rules representing SIMD instructions



Code Selection with SIMD Instructions [Leupers] (1)

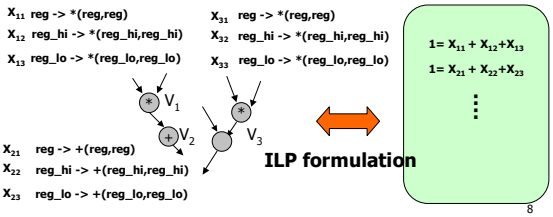
1. Annotating all candidates of rules for each node
2. Formulating constraints into ILP
3. Solving ILP



7

Code Selection with SIMD Instructions [Leupers] (2)

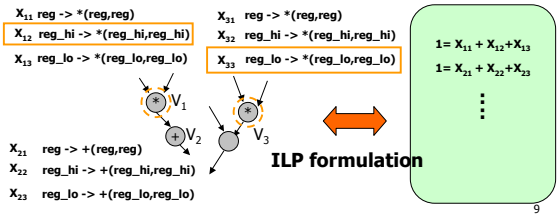
1. Annotating all candidates of rules for each node
2. Formulating constraints into ILP
3. Solving ILP



8

Code Selection with SIMD Instructions [Leupers] (3)

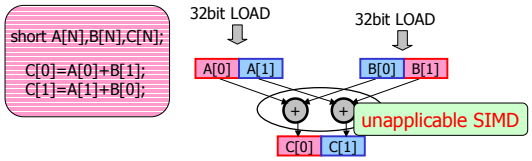
1. Annotating all candidates of rules for each node
2. Formulating constraints into ILP
3. Solving ILP



9

Limitation of Leupers' Method

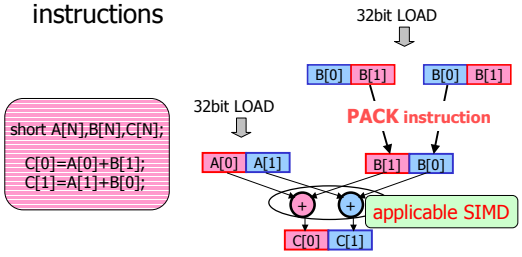
- Leupers' method does not consider PACK instructions



10

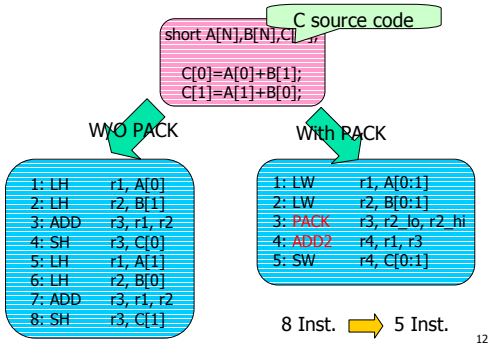
Introduction of PACK Instructions

- PACK instructions help to utilize SIMD instructions



11

Improved Code by PACK Instructions



12

Outline

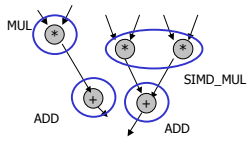
- Introduction
- Previous work (Leupers' method)
  - Code selection with SIMD instructions
- Proposed method
  - Code selection with SIMD and PACK instructions
- Experimental results
- Conclusion

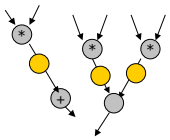
13

Key Idea

- Making it possible to represent PACK/UNPACK operations on DFT

General DFT | Extended DFT



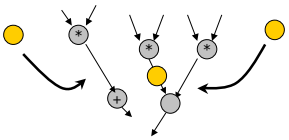


Insert nodes representing data relocation

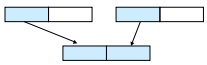
14

Outline of proposed method

1. Additional nodes insertion

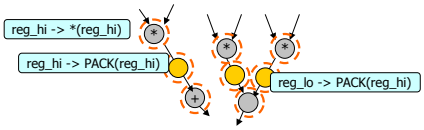


2. PACK, UNPACK rules



PACK : reg_hi -> PACK(reg_hi)
PACK : reg_lo -> PACK(reg_hi)

3. ILP Formulation and code selection

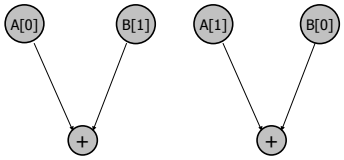


15

Insertion of PACK/UNPACK Operations (1)

```
short A[N],B[N],C[N];

C[0]=A[0]+B[1];
C[1]=A[1]+B[0];
```

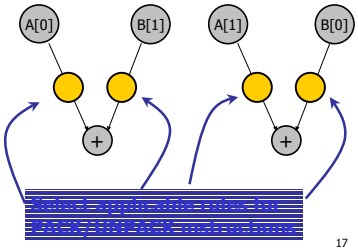


16

Insertion of PACK/UNPACK Operations (2)

```
short A[N],B[N],C[N];

C[0]=A[0]+B[1];
C[1]=A[1]+B[0];
```



17

Rules for PACK Instructions (1)

- Adding rules representing PACK instructions

PACK : reg_hi -> PACK(reg_hi)
PACK : reg_lo -> PACK(reg_hi)
PACK : reg_hi -> PACK(reg_lo)
PACK : reg_lo -> PACK(reg_lo)

PACK : reg_hi -> PACK(reg)
PACK : reg_lo -> PACK(reg)

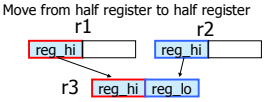
18

Rules for PACK Instructions (2)

- Adding rules representing PACK instructions

PACK : reg_hi -> PACK(reg_hi)
PACK : reg_lo -> PACK(reg_hi)
PACK : reg_hi -> PACK(reg_lo)
PACK : reg_lo -> PACK(reg_lo)

Move from half register to half register



PACKHH r3, r1, r2

PACK : reg_hi -> PACK(reg)
PACK : reg_lo -> PACK(reg)

Move from full register to half register

PACKLL r3, r1, r2

Rules for UNPACK.NOMOVE

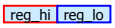
- UNPACK

UNPACK : reg -> UNPACK(reg_lo)
UNPACK : reg -> UNPACK(reg_hi)

UNPACKH r2, r1

- NOMOVE

NOMOVE : reg_hi -> NOMOVE(reg_hi)
NOMOVE : reg_lo -> NOMOVE(reg_lo)
NOMOVE : reg -> NOMOVE(reg)



ILP Formulation

- Constraints in Leupers's method
  - Selection of a single rule
  - Consistency of target nonterminals
  - Common subexpressions
  - Node paring
  - Schedulability
- Additional constraints
  - Node pairing for PACK instructions
  - Packed Data

Outline

- Introduction
- Previous work (Leupers's method)
  - Code selection with SIMD instructions
- Proposed method
  - Code selection with SIMD and PACK instructions
- Experimental results
- Conclusion

Experiments

- Objective
  - Confirming the effectiveness of the proposed method
- Target compilers
  C1: Compiler without SIMD optimization
  C2: Compiler with SIMD optimization
  C3: Compiler with SIMD and PACK optimization
- Instruction set of target processor
  - Subset of DLX(integer arithmetic, jump, branch, compare, load, store), SIMD addition, SIMD multiplication, 4 PACK instructions

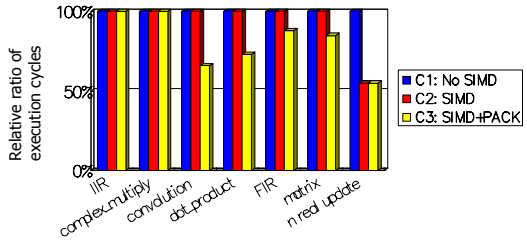
Experiments

- Application
  - DSPstone benchmark
- ILP solver
  - lp_solve*
- Experimental Environment
  - CPU : Intel Pentium. 2.8GHz, Memory : 512MB, OS : RedHat Linux 8.0

*Eindhoven University of Technology : ftp.es.ele.tue.nl/pub/lp_solve/

Results

• Comparison of execution cycles



n 20% reduction in execution cycles

Results

The table of Compilation time, and the number of variables and constraints in ILP

	Leupers's method			Proposed method		
	[sec]	# of Var.	# of Cons.	[sec]	# of Var.	# of Cons.
IIR	0.99	189	190	1.99	2304	7974
complex_multiply	0.09	62	69	0.18	776	1789
convolution	0.09	149	174	1.99	2062	7504
dot_product	0.08	67	88	0.18	704	1522
FIR	0.17	305	627	5679.00	3660	20097
matrix	0.12	21	25	3.79	92	101
n_real_updates	0.12	129	137	22.72	2166	4577

• 5 programs : a few seconds
• n_real_updates : about 20 seconds
• FIR : 5000+ seconds

Outline

- Introduction
- Previous work (Leupers's method)
  - Code selection with SIMD instructions
- Proposed method
  - Code selection with SIMD and PACK instructions
- Experimental results
- Conclusion

Conclusion

- Code selection method with PACK instructions
- 20% reduction in execution cycles
  - Proposed method is effective
- Future work
  - Applying proposed method other processors
  - Developing heuristics that shorten compilation time