Improving CISC Instruction Decoding Performance Using a Fill Unit

Mark Smotherman

Manoj Franklin

Clemson University
Clemson, South Carolina

Overview

1. Instruction decoding and CISC complications
2. Decoupled microarchitectures
3. Fill unit and decoded instruction cache
4. Baseline decoder structure
5. Fill unit decoding and Tree-like line
6. Decoding rate improvements
7. Branch prediction
8. Register renaming
9. Conclusions

Instruction Decoding

- Identify operation and operands
- Rename registers to remove false dependencies
  - Map each use
  - Allocate and map each definition

CISC Complications

- Variable-length instructions
- Implicit register definitions and uses
- Multiple operations per instructions
- Multiple memory accesses per instructions
- Fixed registers (e.g., condition codes)
- Example is PUSH mem
  - Decrement stack pointer (implicit)
  - Load operand from memory
  - Store operand from into stack (implicit)

Decoupled Microarchitectures

- Fixed-length microoperations
- Explicit register definitions and uses
- Multiple microoperations per instruction
- Patt, Melvin, Hwu, Shebanow, and others for HPS-VAX (1985)
  - 4 HPS microoperations per VAX instruction
  - Reduced CPI from 6 to 2
- Approach of Intel Pentium Pro, AMD K5, Motorola 68060

Decoder bandwidth can be a problem

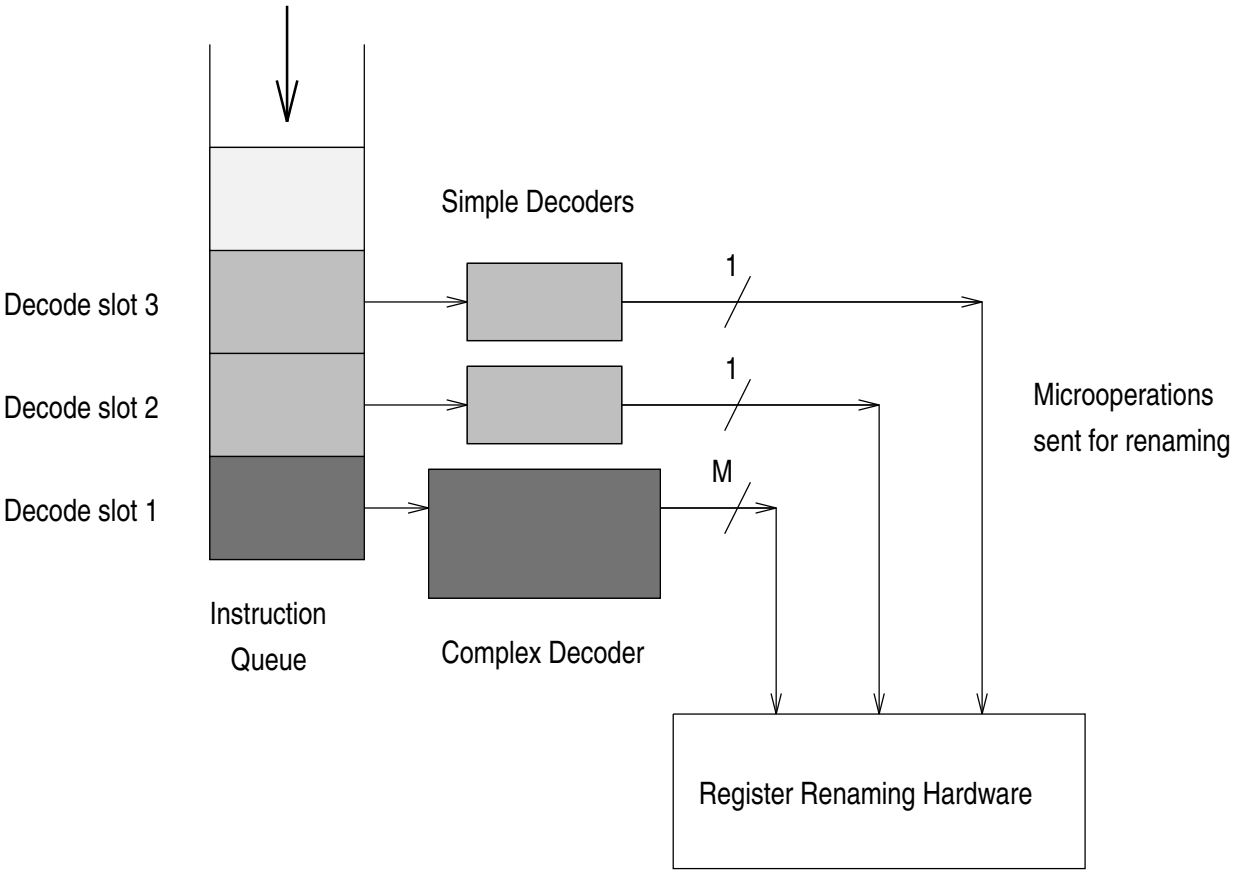
Use fill unit and decoded instruction cache

Extensions

- Tree-like line with microoperations from both paths of conditional branch
- Register renaming scheme for microarchitecture registers

Baseline Decoder Structure

One complex decoder plus two simple decoders

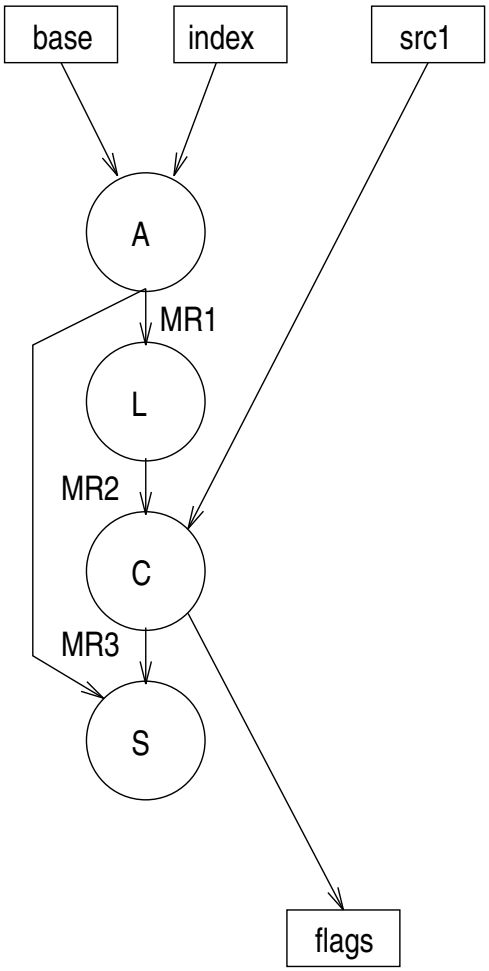


Instruction Classification

- Simple - register-to-register operation
- Complex - memory operand(s)
- mov reg,mem and mov mem,reg may be simple (load and store)

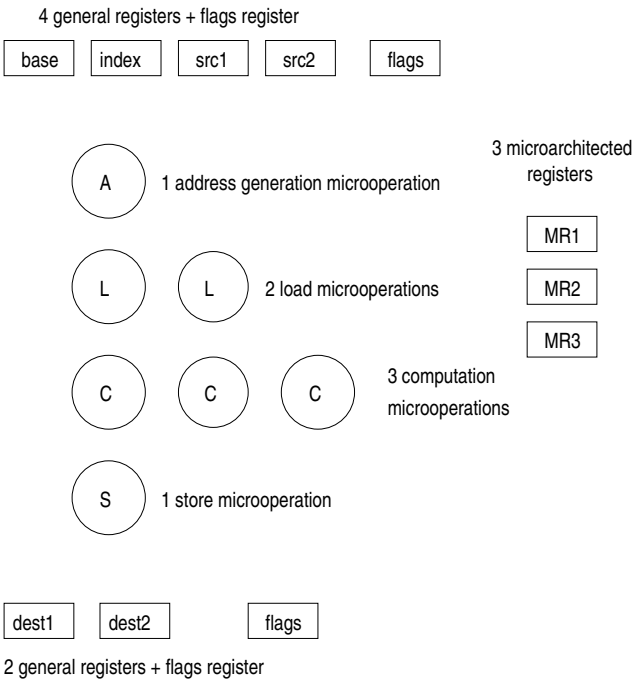
Decoding Example

```
ADD [EBX+EAX],ECX  MR1 <- EBX + EAX
                   MR2 <- Mem[MR1]
                   Flags, MR3 <- MR2 + ECX
                   Mem[MR1] <- MR3
```

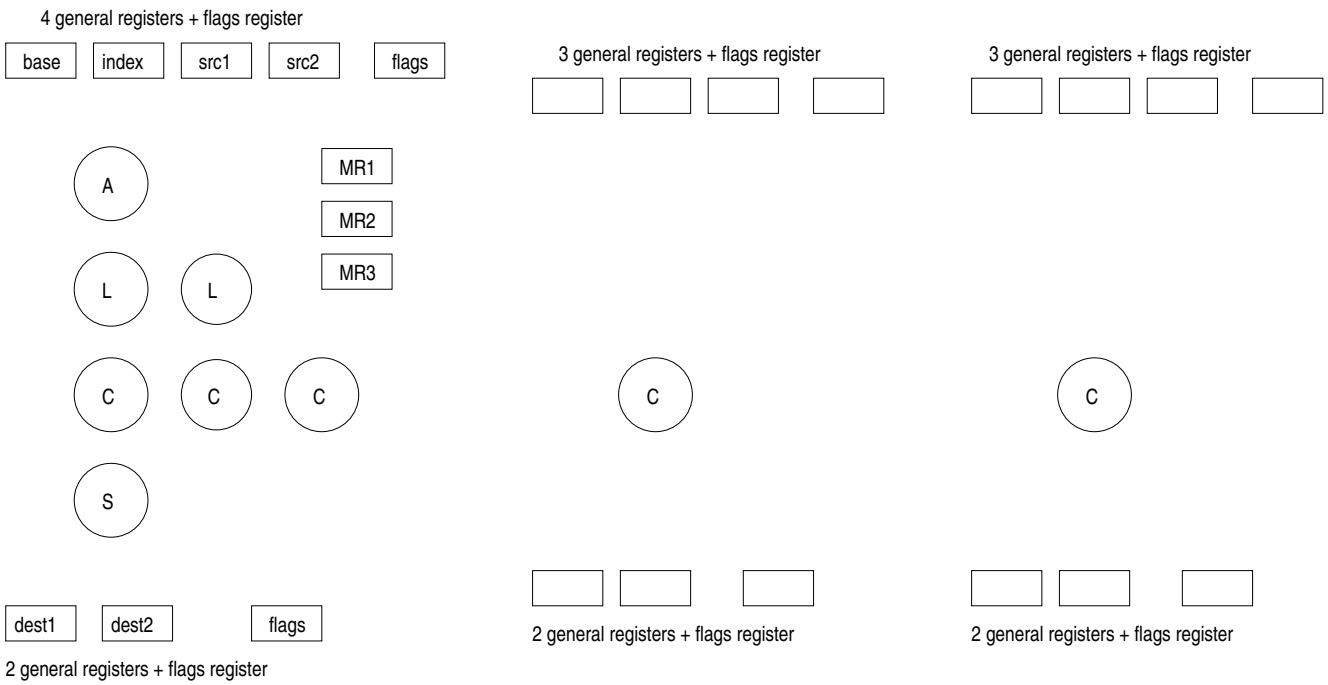


Resource Templates

Complex instruction



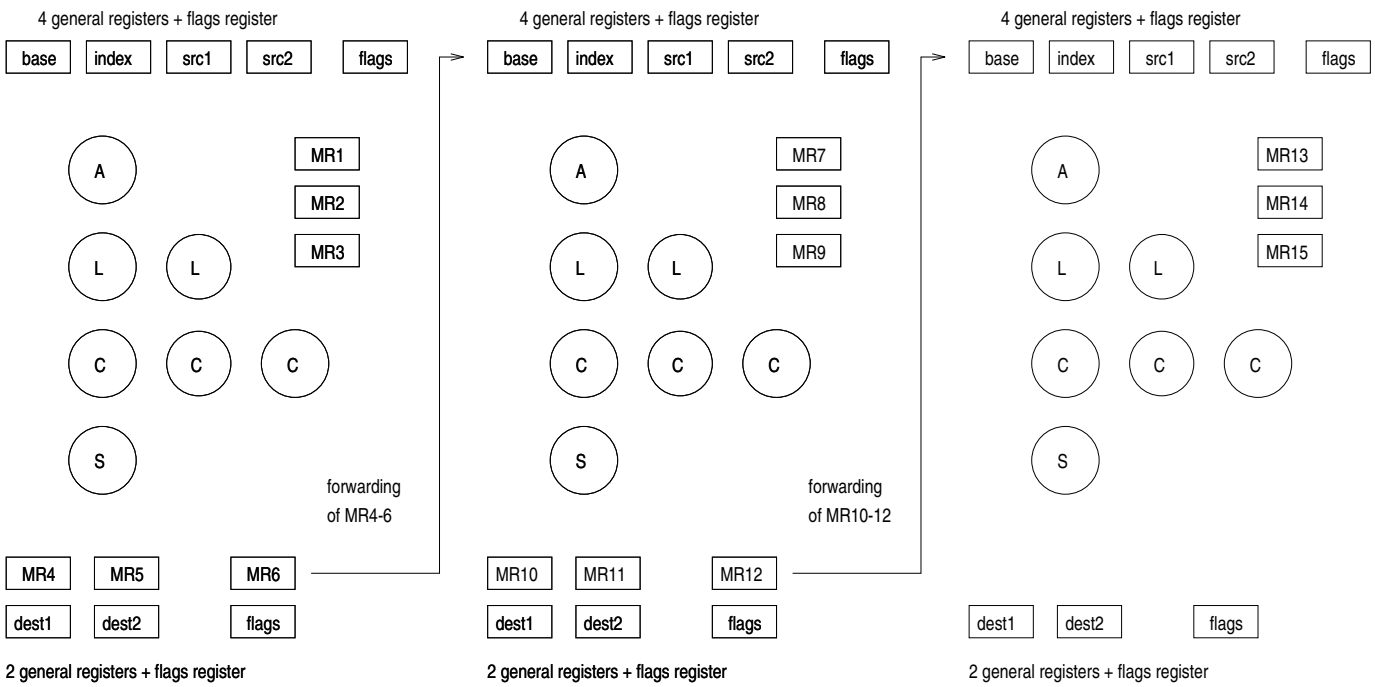
Complex plus two simple instructions



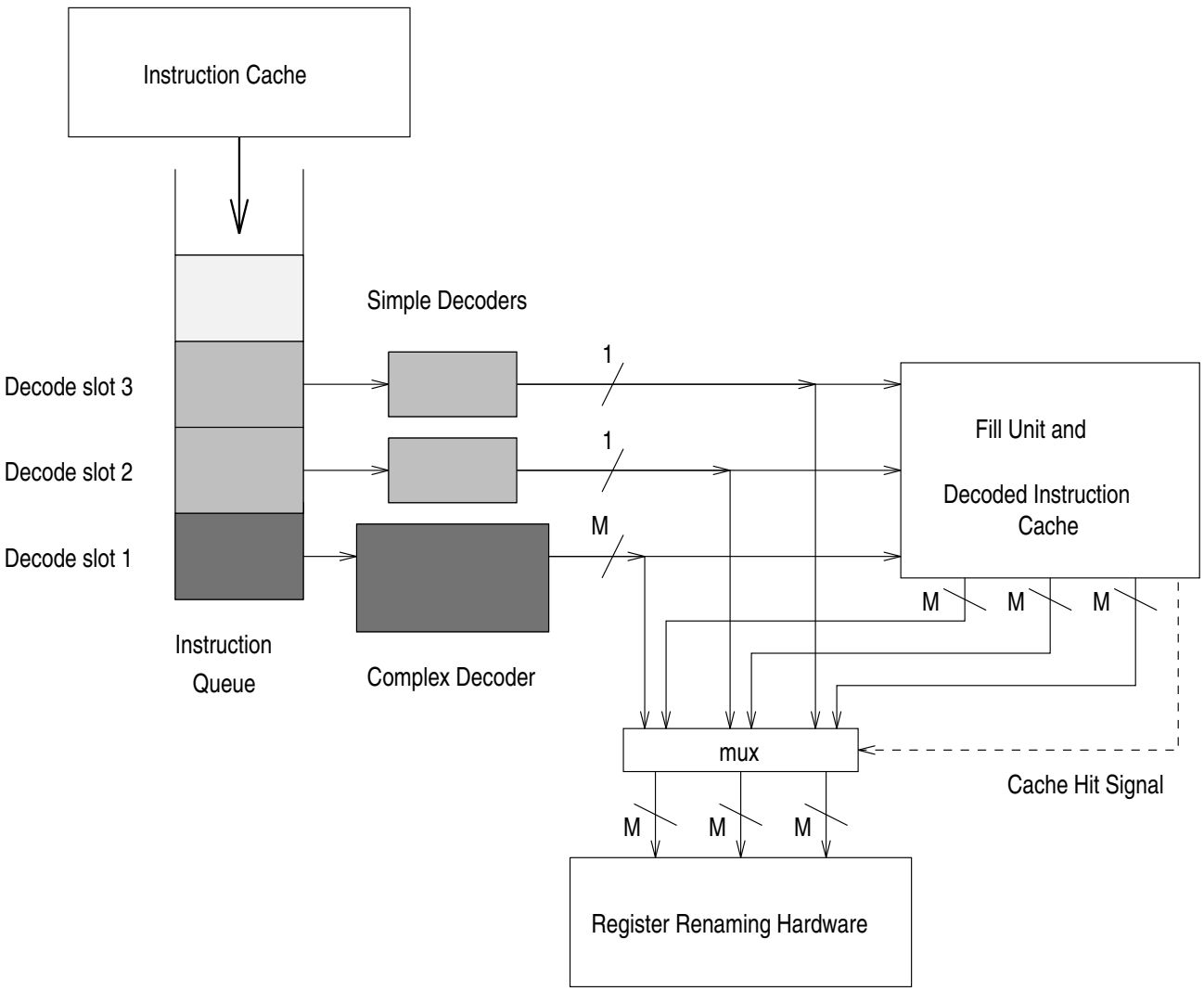
Decoded Instruction Cache

- Collect up to three instructions (21 microoperations)
- Complex doesn't have to be first
- Line must begin and end on instruction boundary
- Stop filling on conditional branch, return, or indirect jump
- Disable DIC in exception recovery mode

Up to three complex instructions per line



Fill Unit Decoding



Tree-Like Line

- Collect both paths of conditional branch
- Branch doesn't have to be first
- Predict which set of microoperations
- Stop filling on second branch, return, or indirect jump
- Example

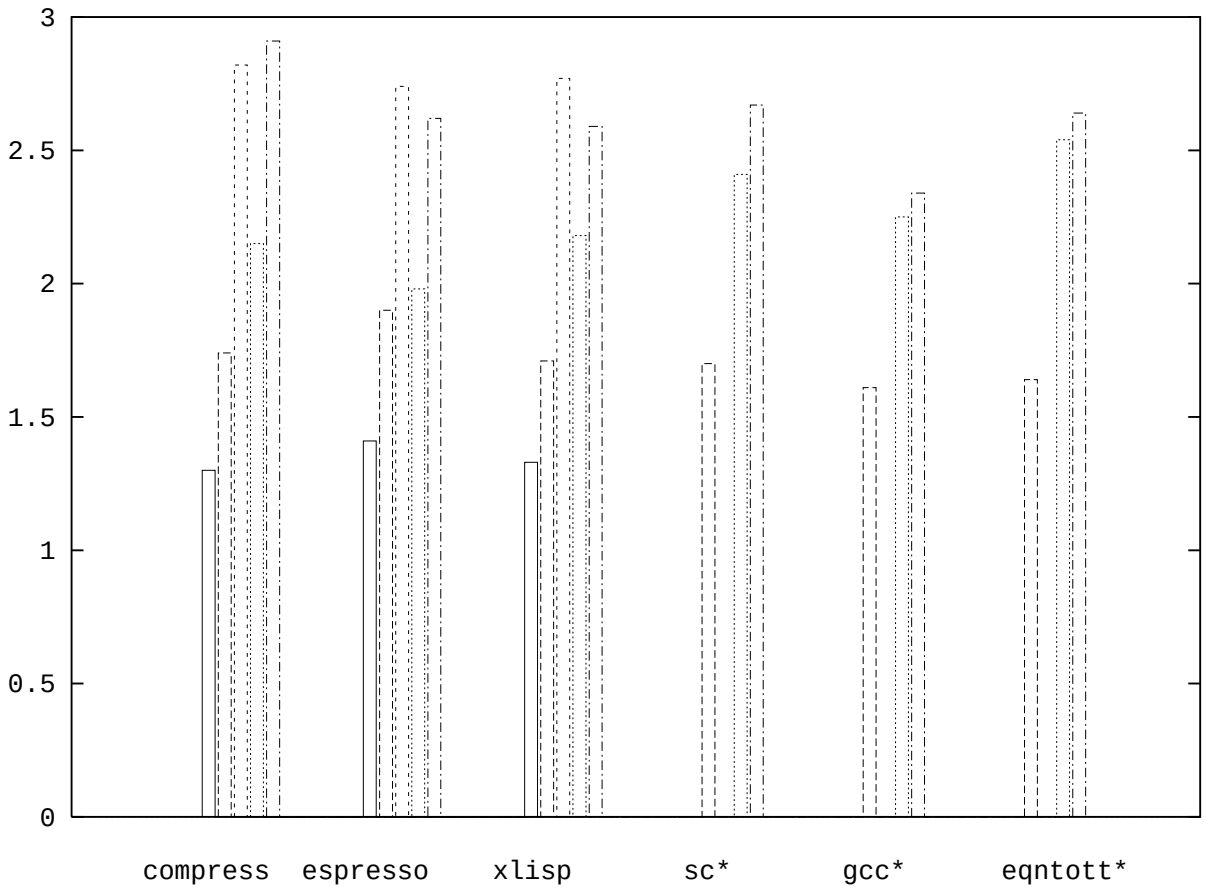
```
x    instruction
x+1  conditional branch to address y
x+2  untaken_instruction_0
x+3  untaken_instruction_1
x+4  untaken_instruction_2
...
y    taken_instruction_0
y+1  taken_instruction_1
y+2  taken_instruction_2
y+3  taken_instruction_3
```

(x)	(x+1)	(x+2)	untaken
(x)	(x+1)	(y)	taken

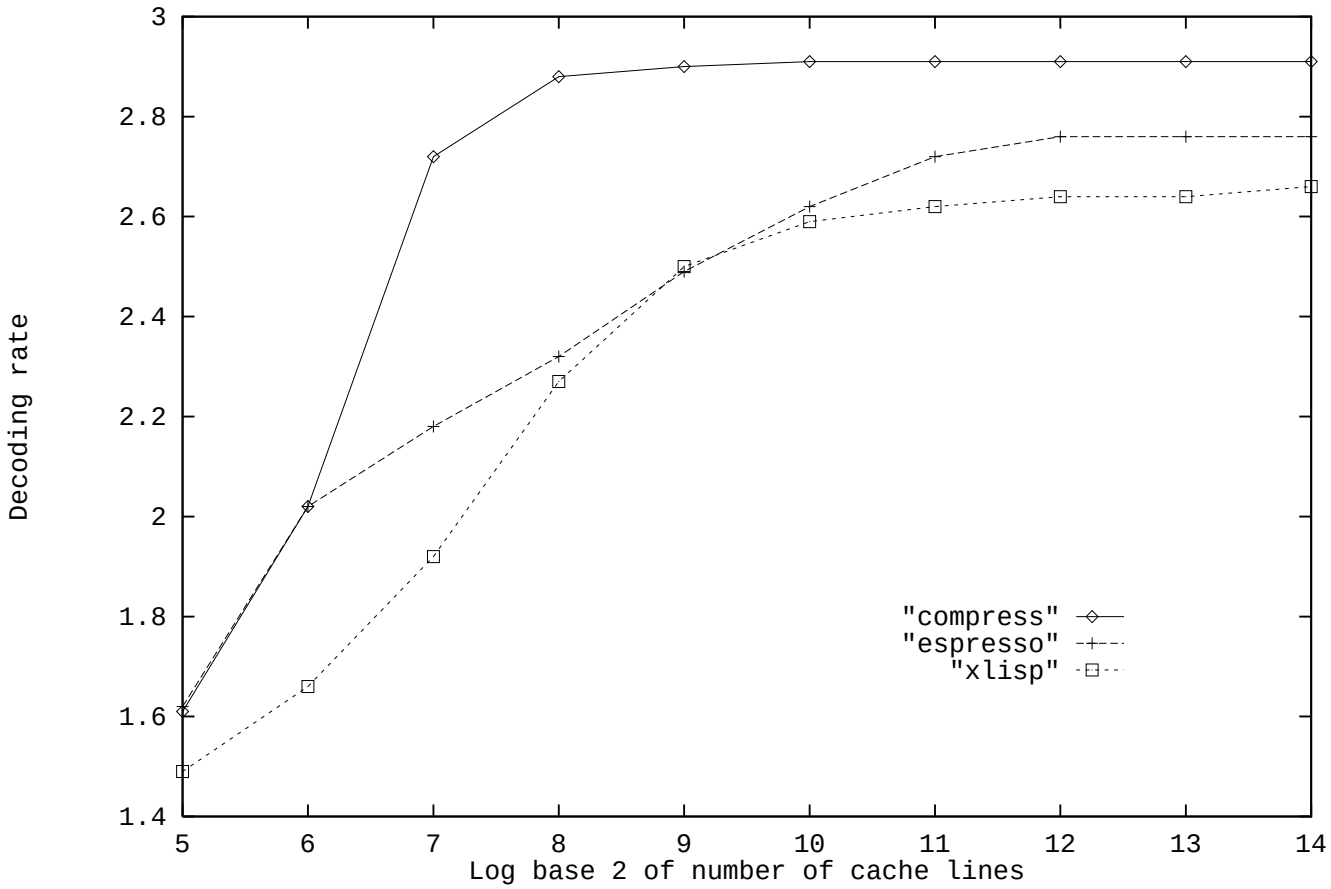
(x+1)	(x+2)	(x+3)	untaken
(x+1)	(y)	(y+1)	taken

Decoding Rate Improvement with Fill Unit

Register-to-register as simple
Register-to-register plus moves as simple *
Three complex decoders
Basic fill unit *
Tree-like line *



Decoding Rate versus DIC Size



Branch Prediction

- Branch may not be first instruction in line
- Multiple instructions per path
- Example

```
x    instruction
x+1  conditional branch to address y
x+2  untaken_instruction_0
x+3  untaken_instruction_1
x+4  untaken_instruction_2
...
y    taken_instruction_0
y+1  taken_instruction_1
y+2  taken_instruction_2
y+3  taken_instruction_3
```

  - Normal BTB entry: (x+1 : y)
  - DIC BTB entries: (x : y+2, x+3) and/or (x+1 : y+2, x+4)

Register Renaming

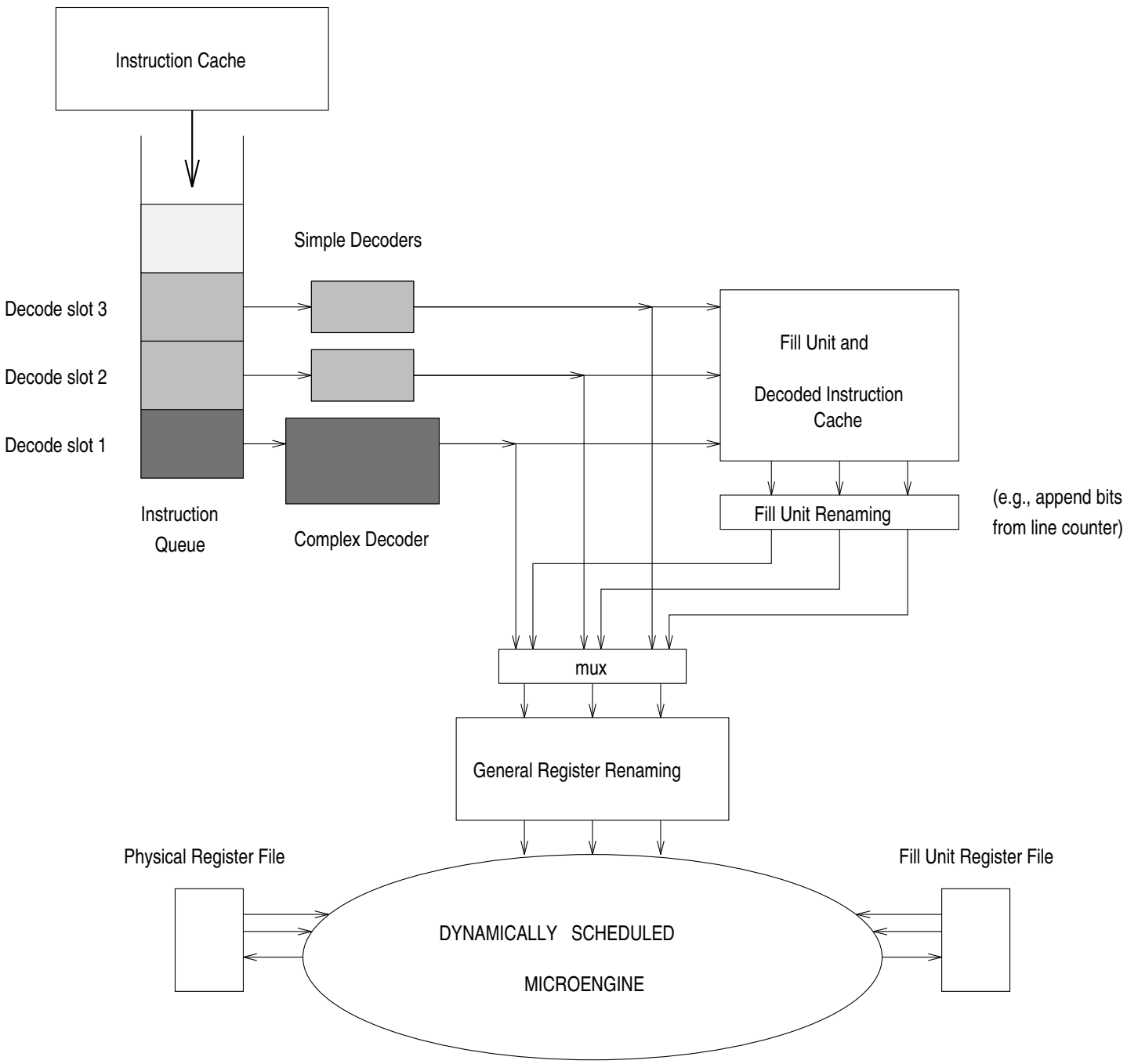
- Higher decoding rate impacts renaming rate

```
ADD    EAX, [EBX+ESI]  MR1        <- EBX + ESI
                       MR2        <- memory[MR1]
                       EAX, Flags <- EAX + MR2
ADD    EAX, 10         EAX, Flags <- EAX + 10
ADD    EBX, 4          EBX, Flags <- EBX + 4

use EBX, ESI   def MR1
use MR1        def MR2
use EAX, MR2   def EAX, Flags
use EAX        def EAX, Flags
use EBX        def EBX, Flags
```

- Atomic issue of DIC line
  - Remove multiple definitions (especially Flags!)
  - Rename microarchitecture registers within line
  - Use access counter of DIC lines (depth of reorder buffer limits the number of microoperations that can be present)

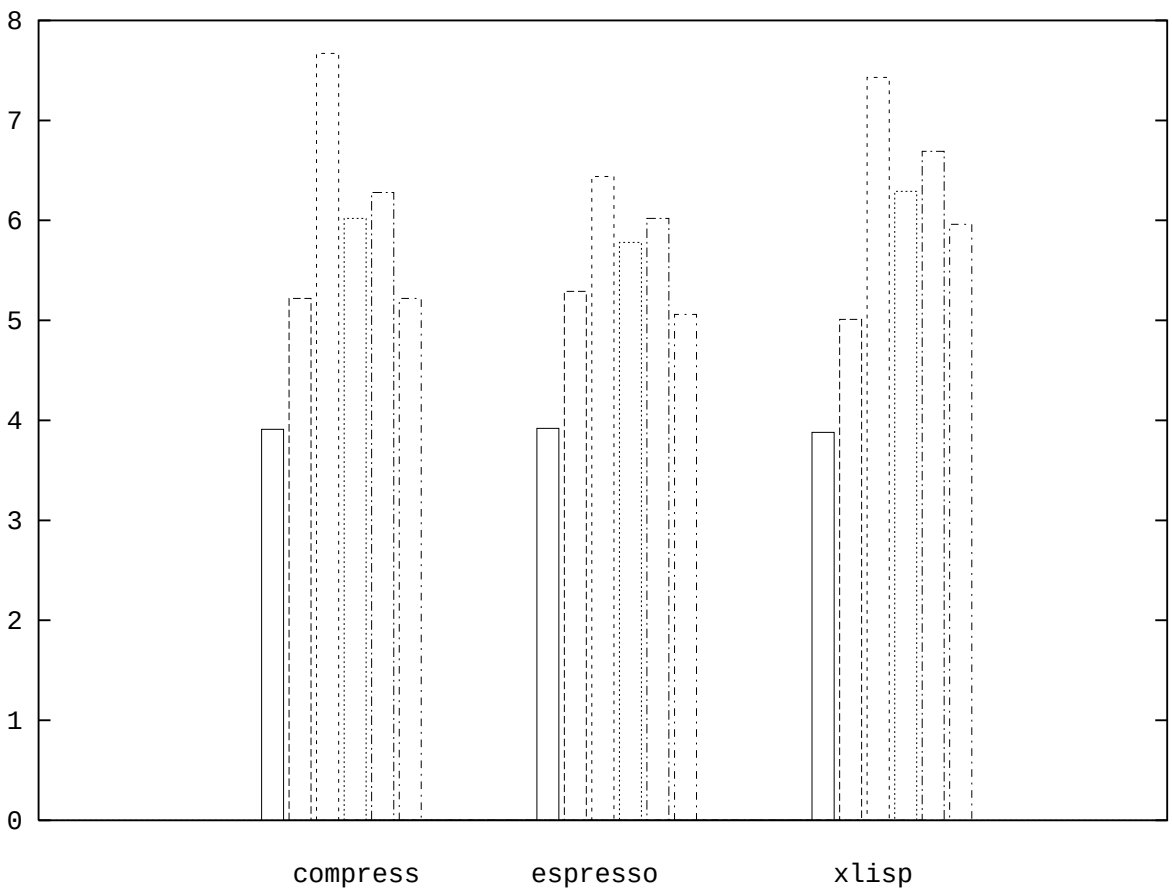
```
ADD    EAX, [EBX+ESI]  MR1        <- EBX + ESI
                       MR2        <- memory[MR1]
                       MR3        <- EAX + MR2
ADD    EAX, 10         EAX        <- MR3 + 10
ADD    EBX, 4          EBX, Flags <- EBX + 4

use EBX, ESI, EAX, EBX  def EAX, EBX, Flags
```

Fill Unit Renaming



Average Renaming Counts of 1K-Entry DIC

Register-to-register as simple
Register-to-register plus moves as simple
Three complex decoders
Basic fill unit
Tree-like line
Atomic issue



Conclusions

- CISC decoding is complex
- CISC decoding needs high bandwidth especially for decoupled microarchitectures
- Decoded instruction cache reduces pressure on decoder
- Tree-like line is an important extension
- Special register file for microarchitecture registers reduces pressure on renaming