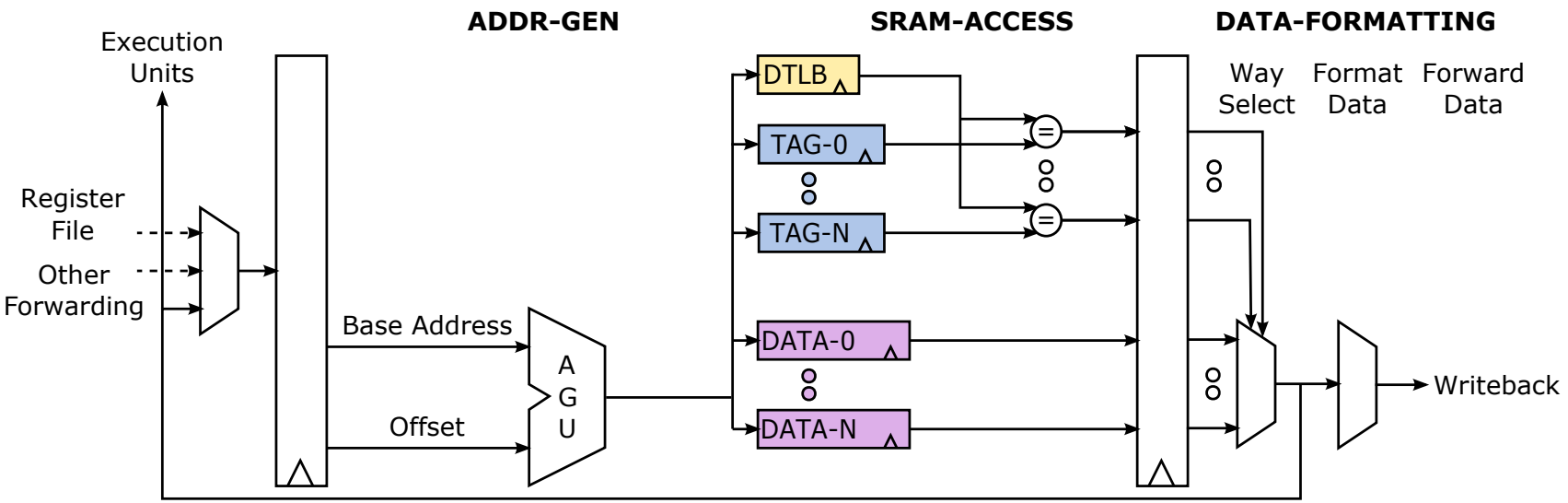
Improving Data Access Efficiency by Using Context-Aware Loads and Stores

Alen Bardizbanyan, Magnus Själander[†], David Whalley[‡], Per Larsson-Edefors

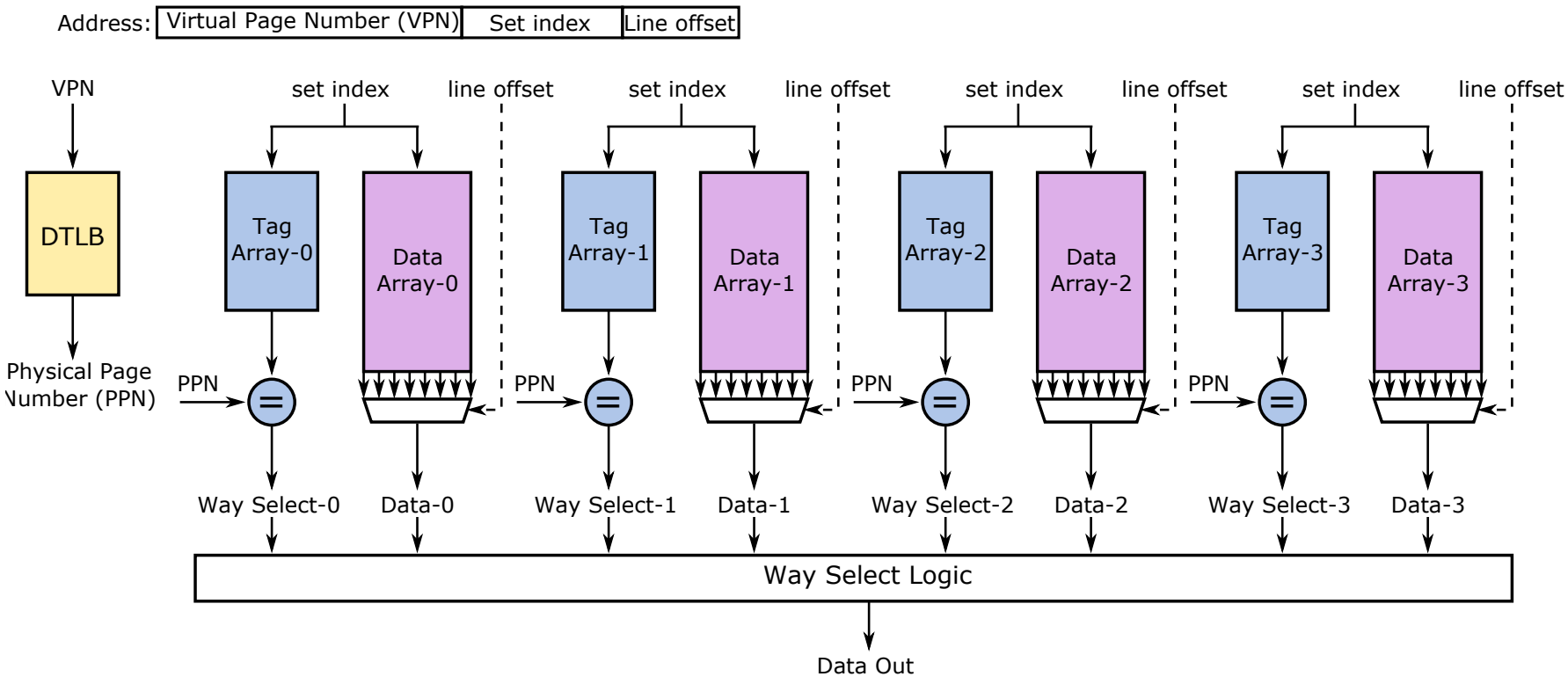
Chalmers University of Technology
[†]Uppsala University
[‡]Florida State University

UPPSALA
UNIVERSITET

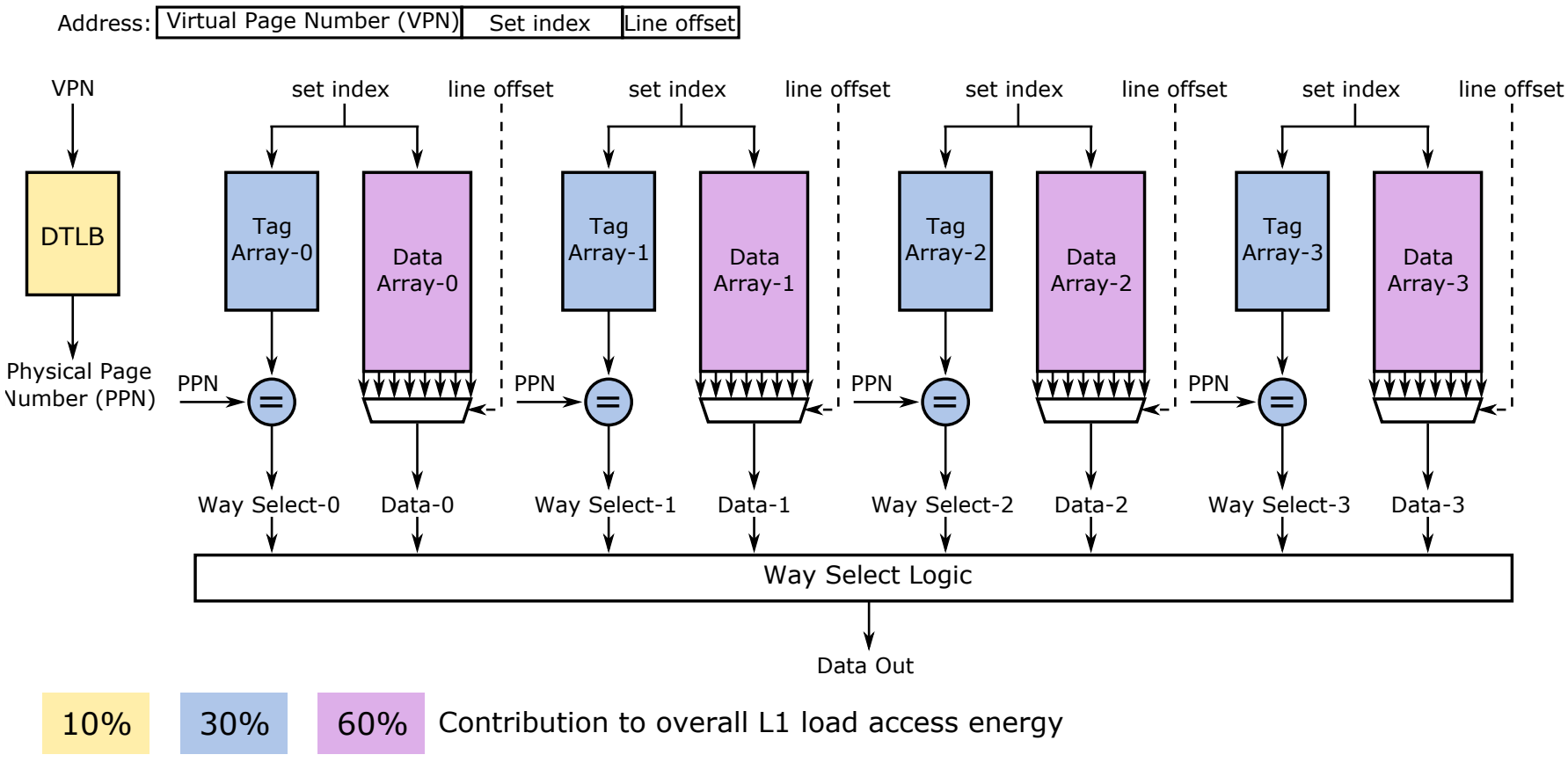
Conventional L1 DC Access



Energy Usage of a 4-way L1 Data Cache

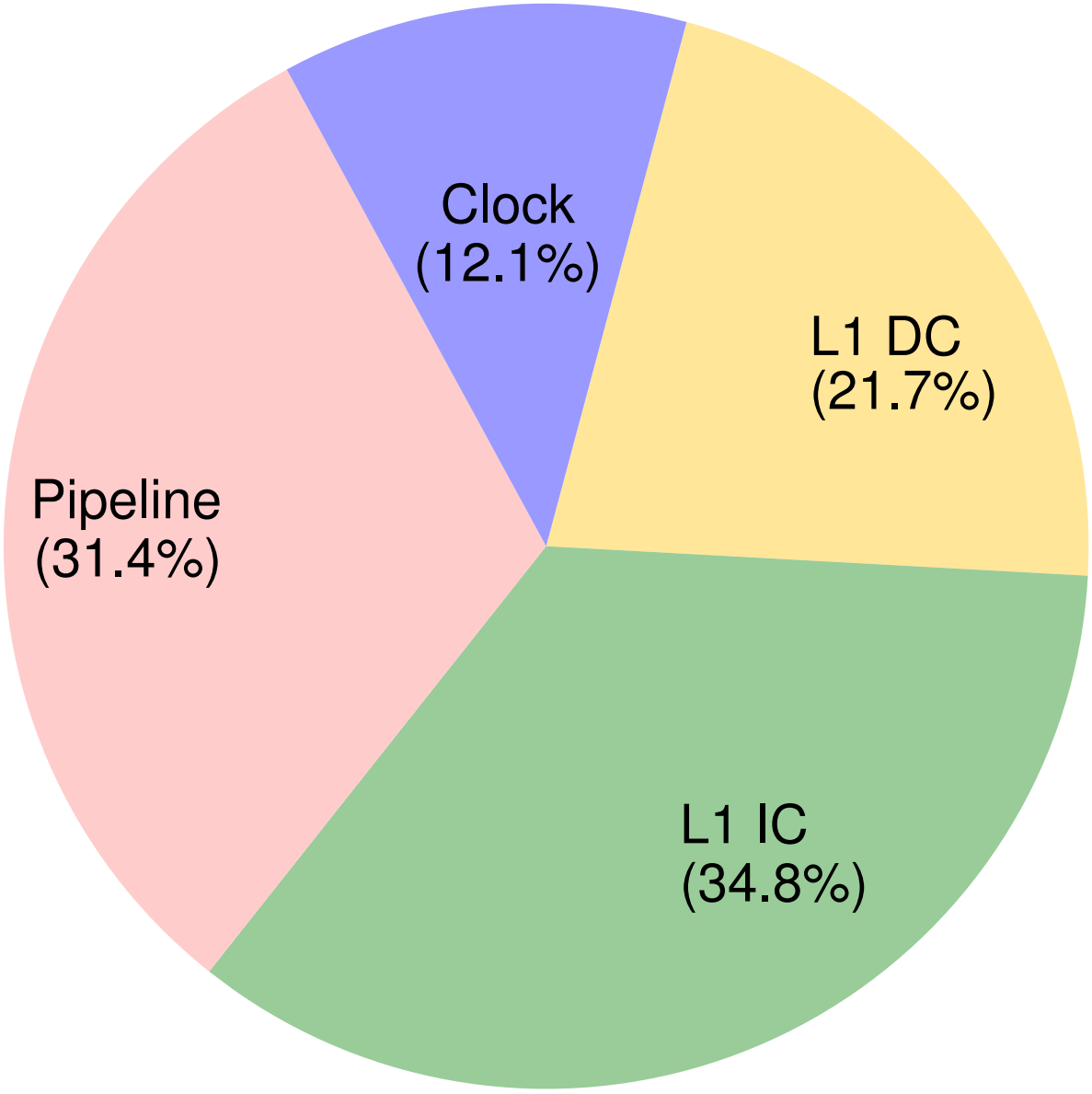


Energy Usage of a 4-way L1 Data Cache

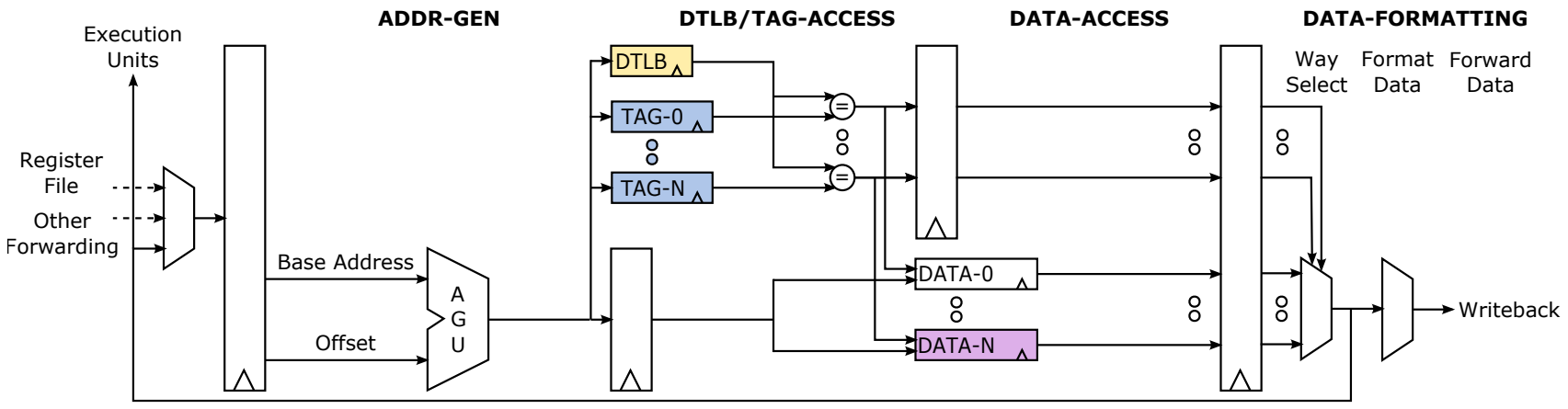


60% of the energy is due to reading the data memories in parallel

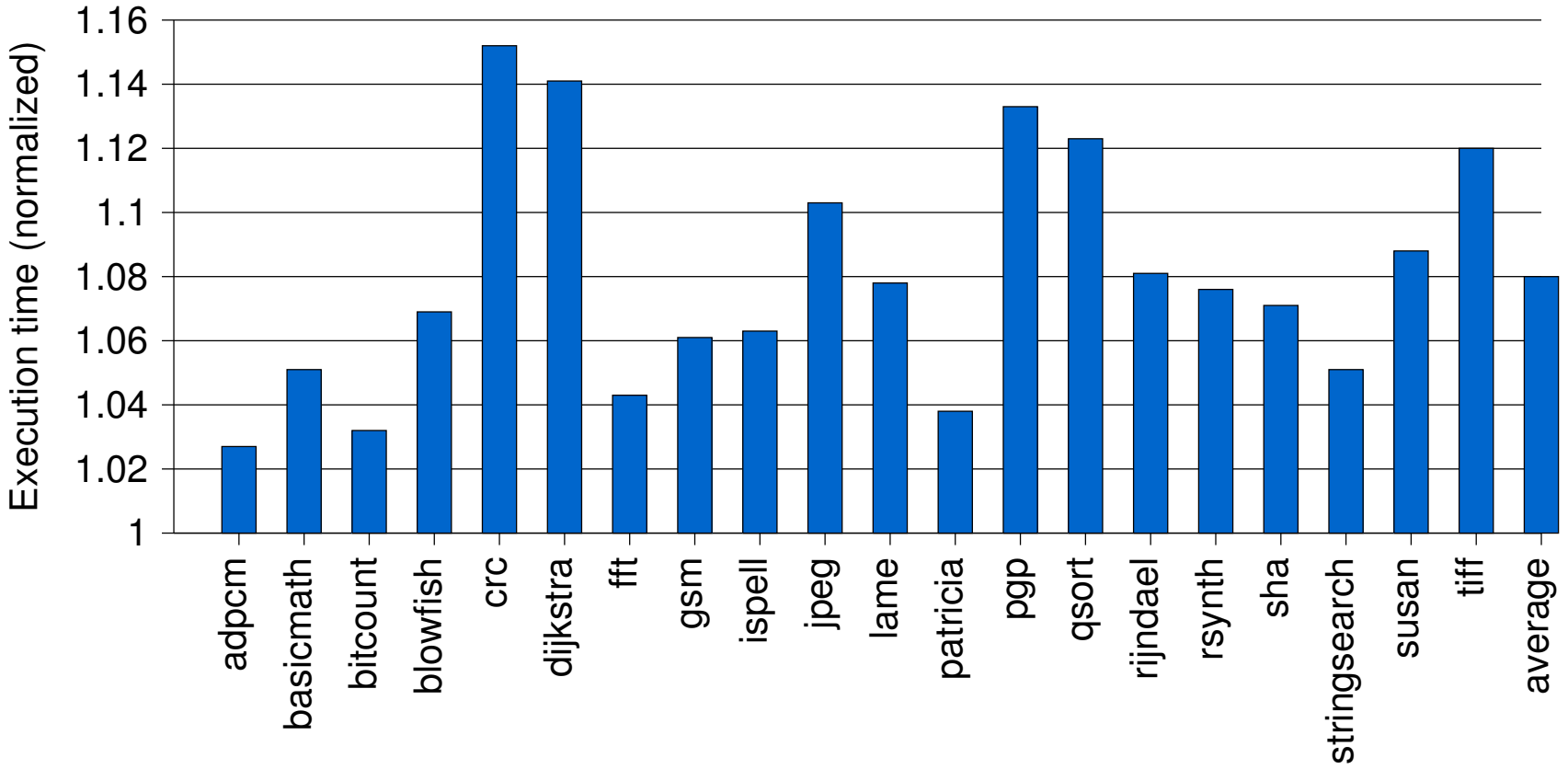
Energy Breakdown of an Embedded Processor



Phased L1 DC Access

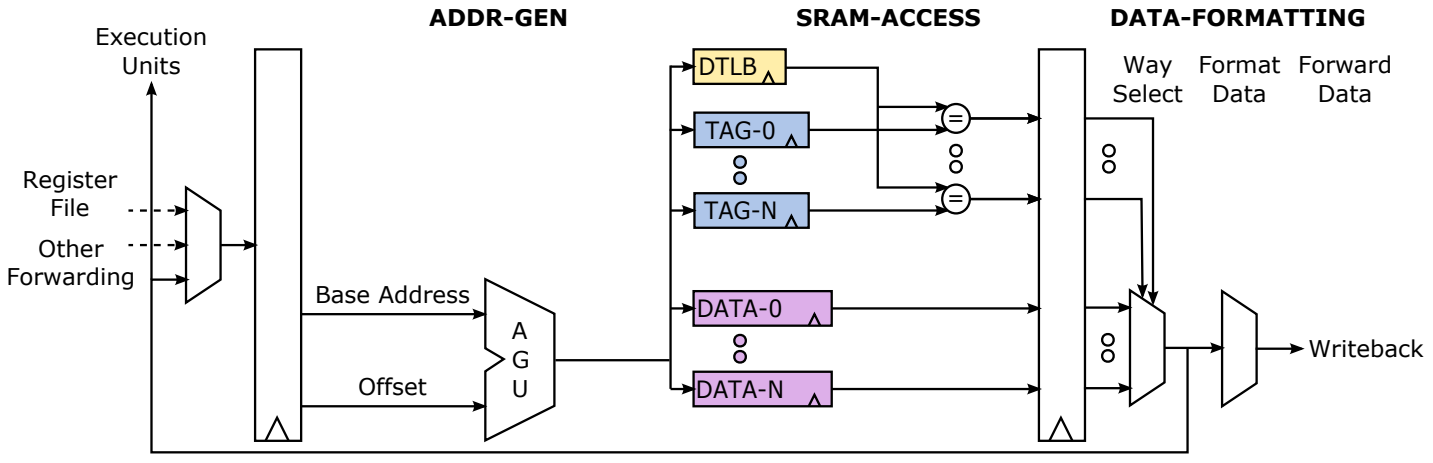


Phased L1 DC Performance Overhead



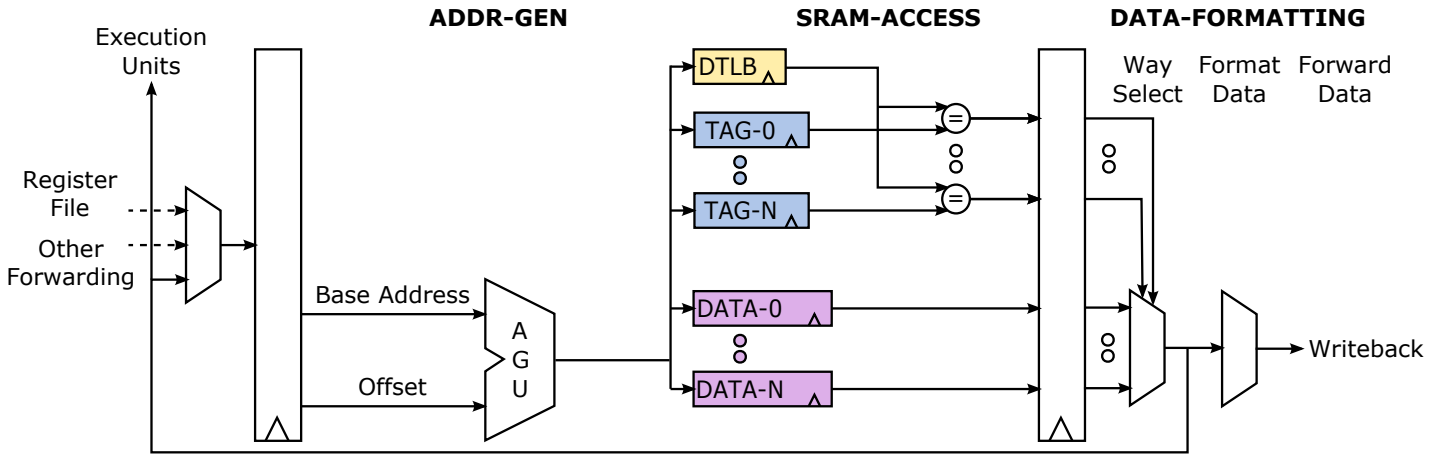
Average performance overhead of 8%.

Context Aware Loads — Case0



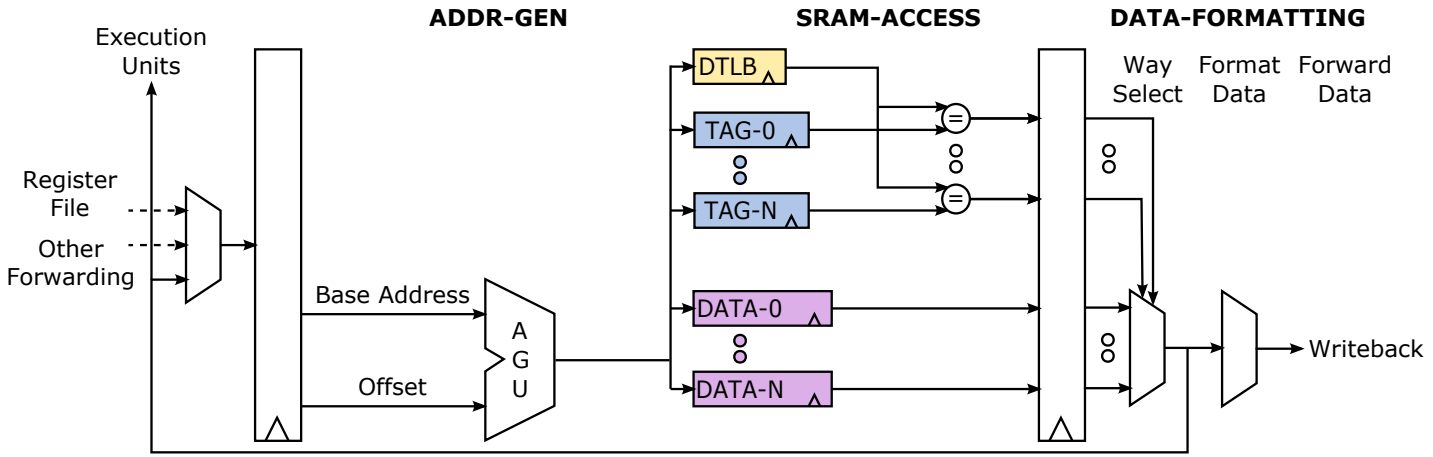
r[2] = M[r[4]+76]
r[3] = r[3]+r[2]

Context Aware Loads — Case0



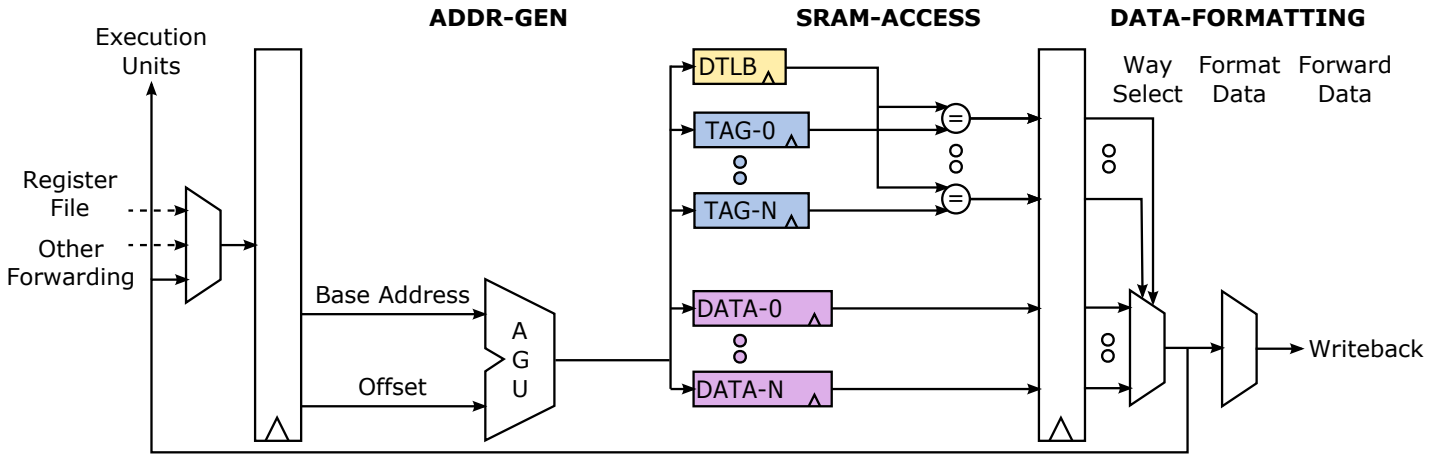
r[2] = M[r[4]+76]
r[3] = r[3]+r[2]

Context Aware Loads — Case0



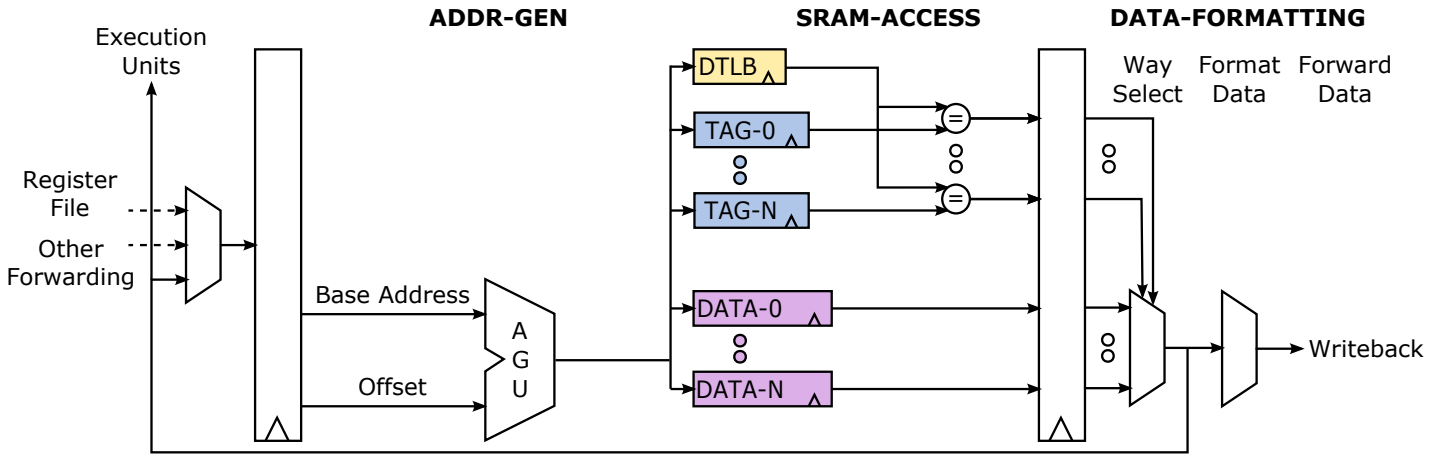
r[2] = M[r[4]+76]
r[3] = r[3]+r[2]

Context Aware Loads — Case1



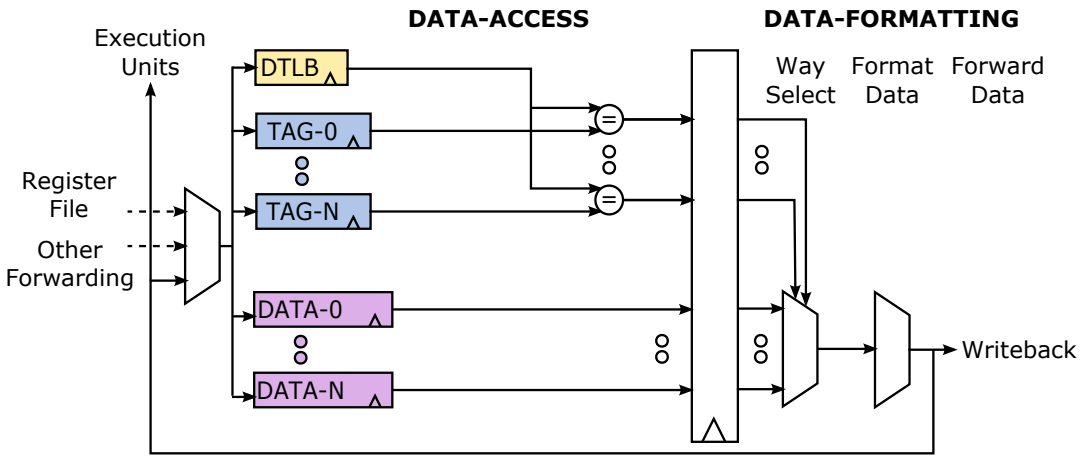
r[2] = M[r[4]]
r[3] = r[3]+r[2]

Context Aware Loads — Case1



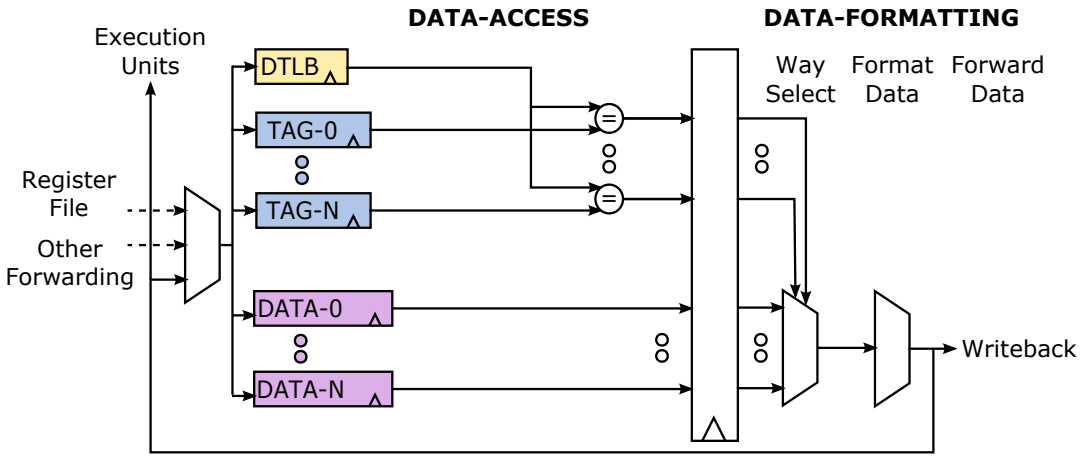
r[2] = M[r[4]]
r[3] = r[3]+r[2]

Context Aware Loads — Case1



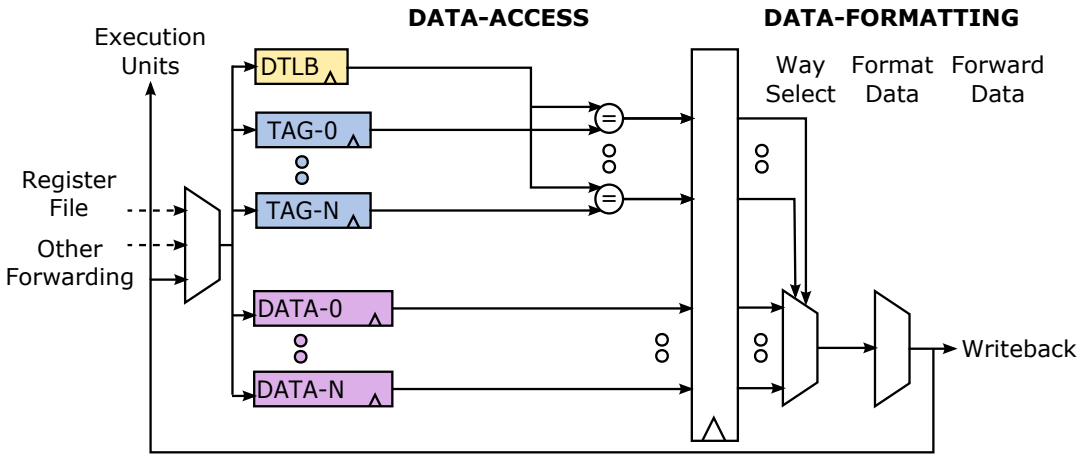
r[2] = M[r[4]]
r[3] = r[3]+r[2]

Context Aware Loads — Case2



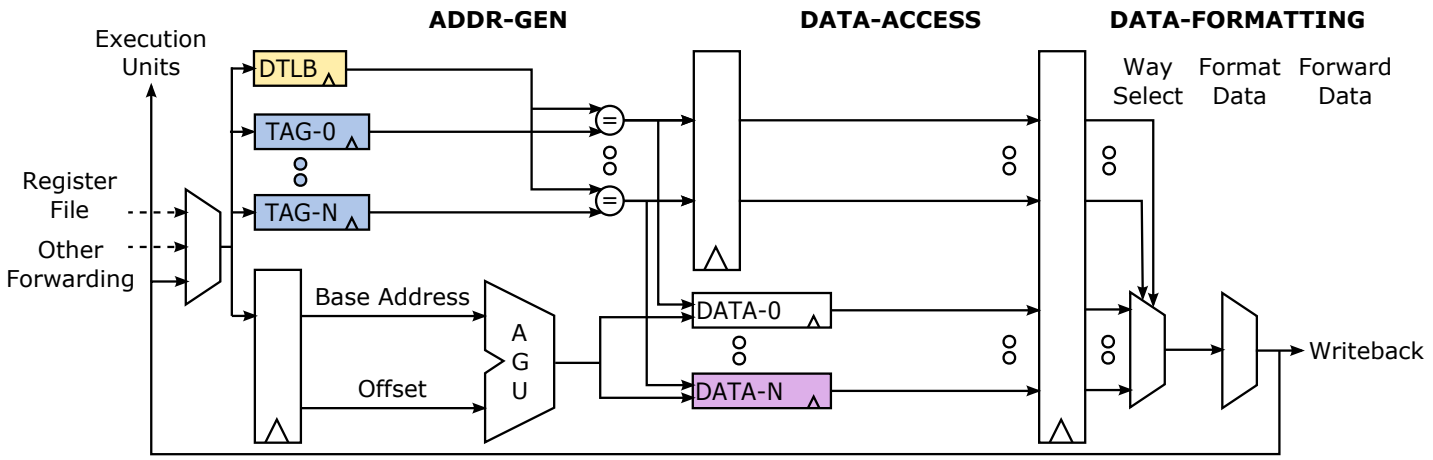
r[2] = M[r[4]+4]
r[3] = r[3]+r[2]

Context Aware Loads — Case2



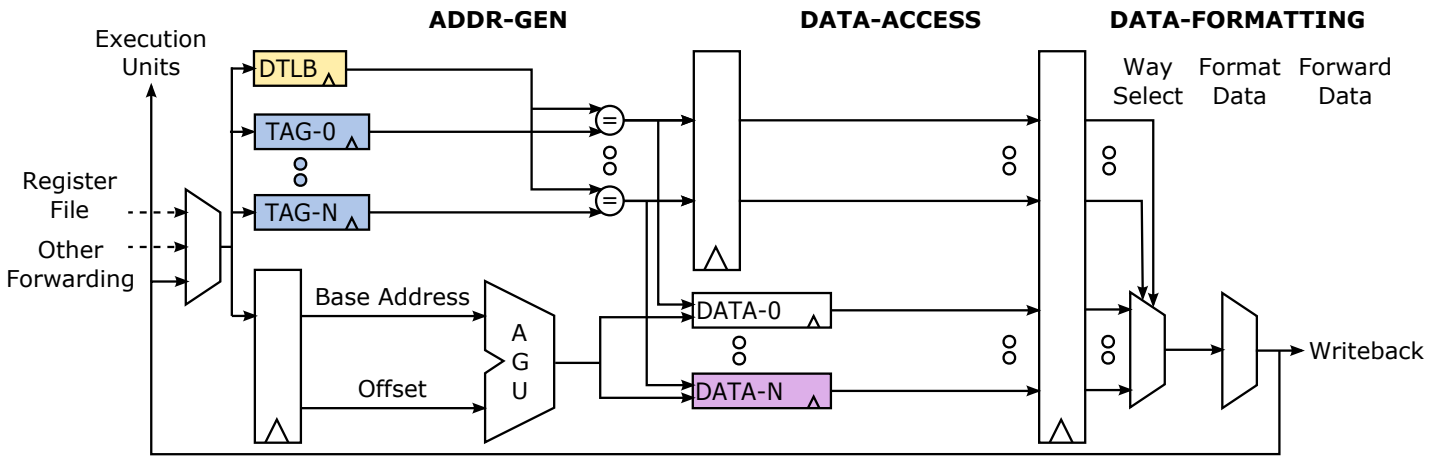
r[2] = M[r[4]+4]
r[3] = r[3]+r[2]

Context Aware Loads — Case2



r[2] = M[r[4]+4]
r[3] = r[3]+r[2]

Context Aware Loads — Case3

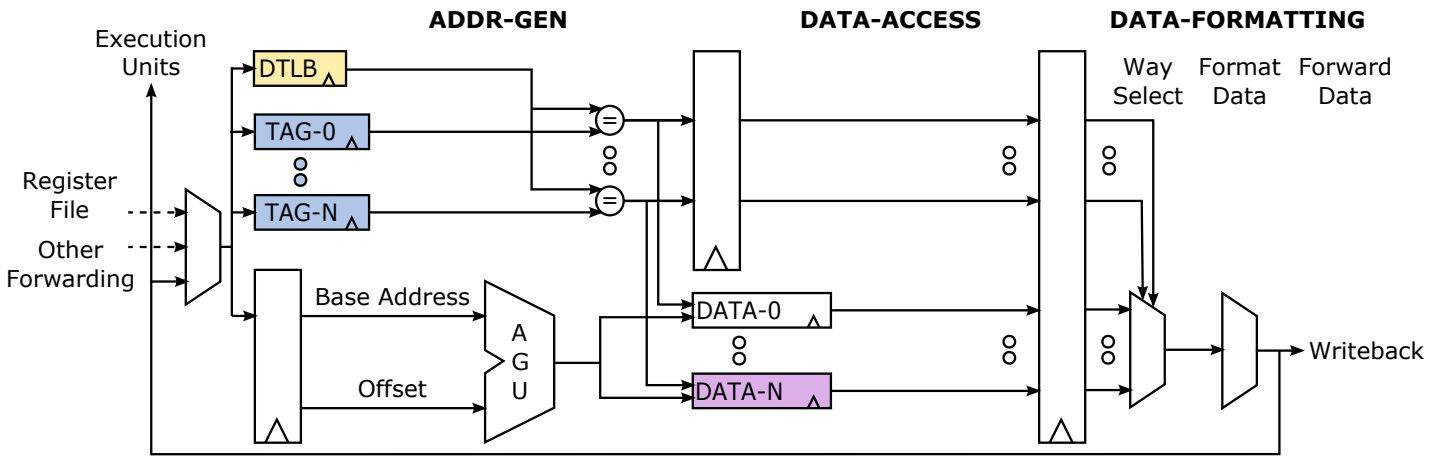


r[2] = M[r[4]+4]

<3 or more insts>

r[3] = r[3]+r[2]

Context Aware Loads — Case3

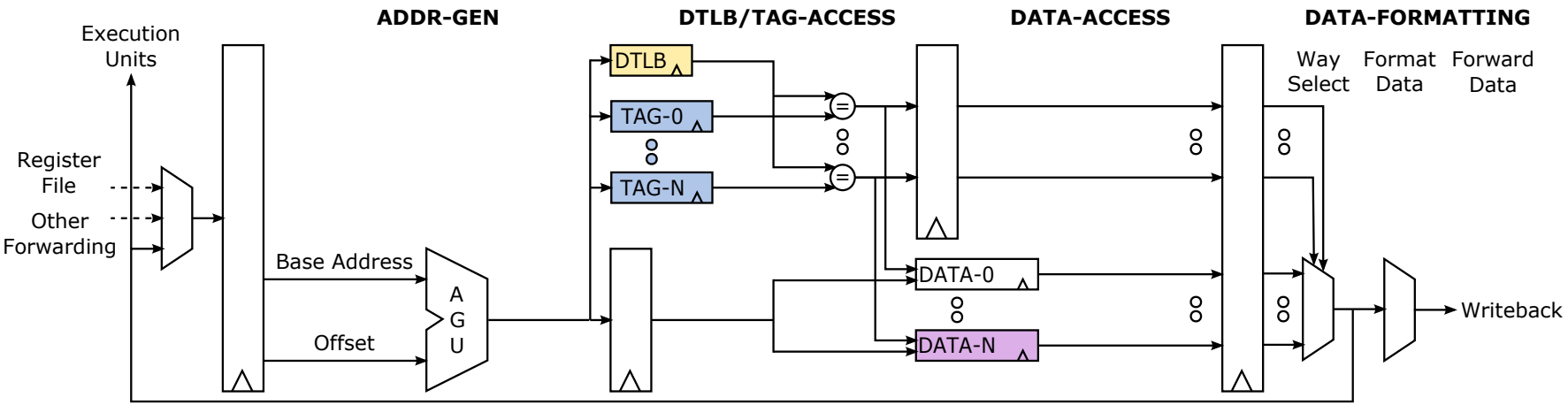


r[2] = M[r[4]+4]

<3 or more insts>

r[3] = r[3]+r[2]

Context Aware Loads — Case3



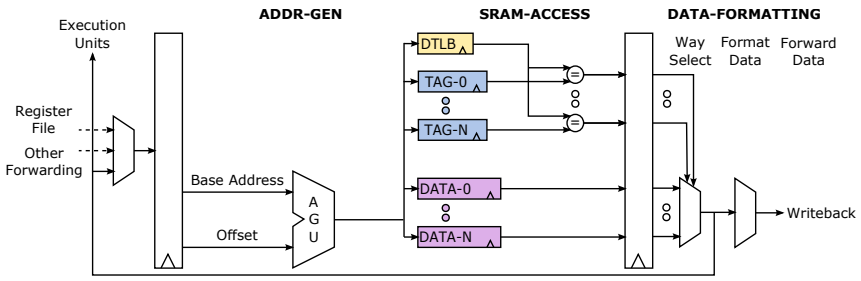
r[2] = M[r[4]+4]

<3 or more insts>

r[3] = r[3]+r[2]

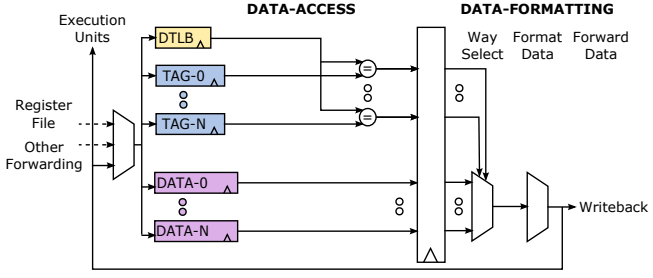
Context Aware Loads — Cases 0-3

r[2] = M[r[4]+76]
r[3] = r[3]+r[2]



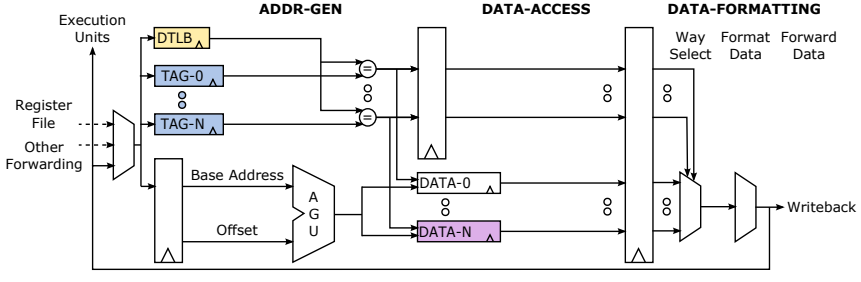
Case0:
Normal Access

r[2] = M[r[4]]
r[3] = r[3]+r[2]



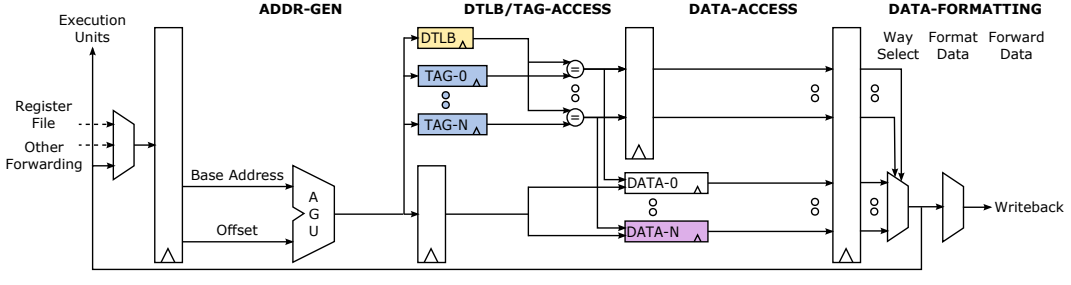
Case1:
Avoids Stalls

r[2] = M[r[4]+4]
r[3] = r[3]+r[2]



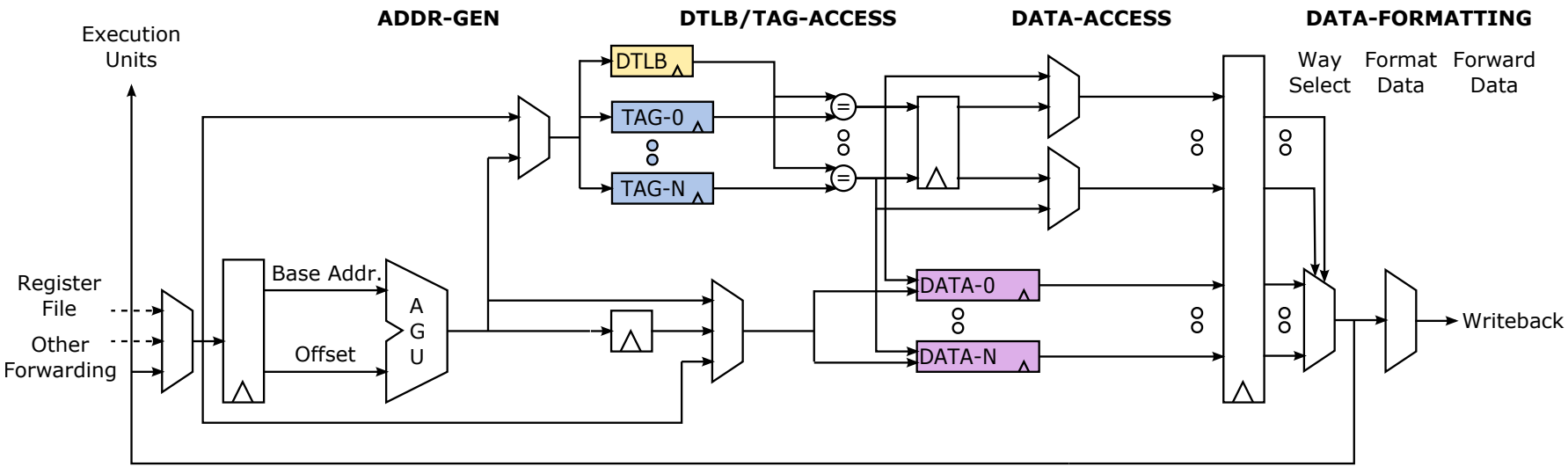
Case2:
1x Data Array Access

r[2] = M[r[4]+4]
<3 or more insts>
r[3] = r[3]+r[2]



Case3:
1x Data Array Access
No Tag Speculation

Context Aware Loads — Pipeline



Strided Accesses

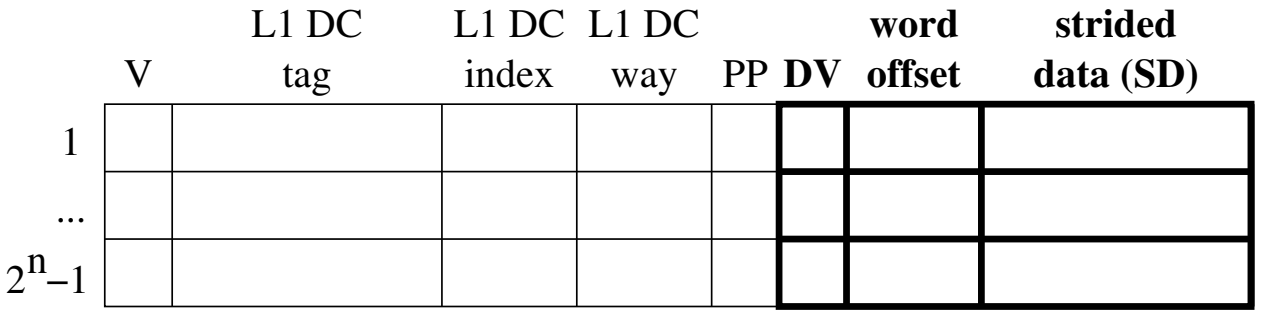
```
L3: r[2]=M[r[4]];
    ...
    r[4]=r[4]+4;
    PC=r[4]!=r[5],L3;
```

```
…
r[22]=M[r[sp]+100];
r[21]=M[r[sp]+96];
r[20]=M[r[sp]+92];
…
```

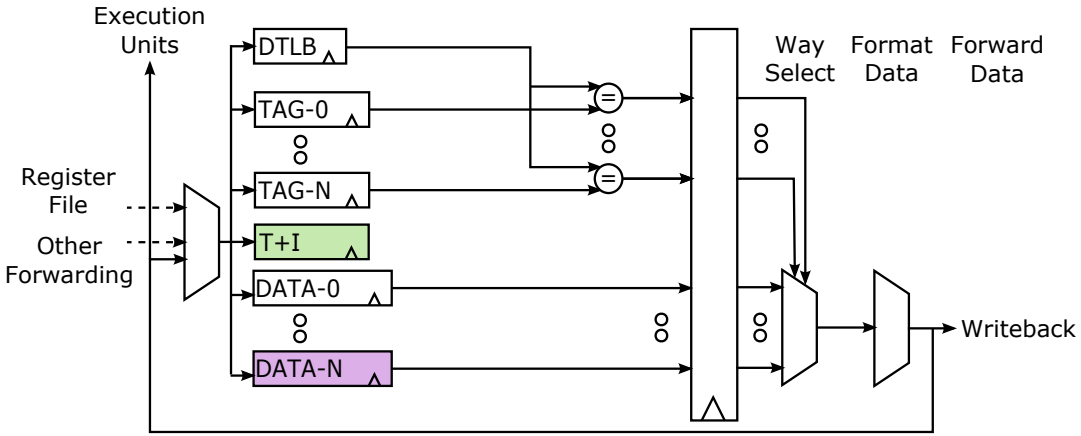
Strided Accesses — Strided Access Structure

```
L3: r[2]=M[r[4]];
    ...
    r[4]=r[4]+4;
    PC=r[4]!=r[5],L3;
```

```
...
r[22]=M[r[sp]+100];
r[21]=M[r[sp]+96];
r[20]=M[r[sp]+92];
...
```



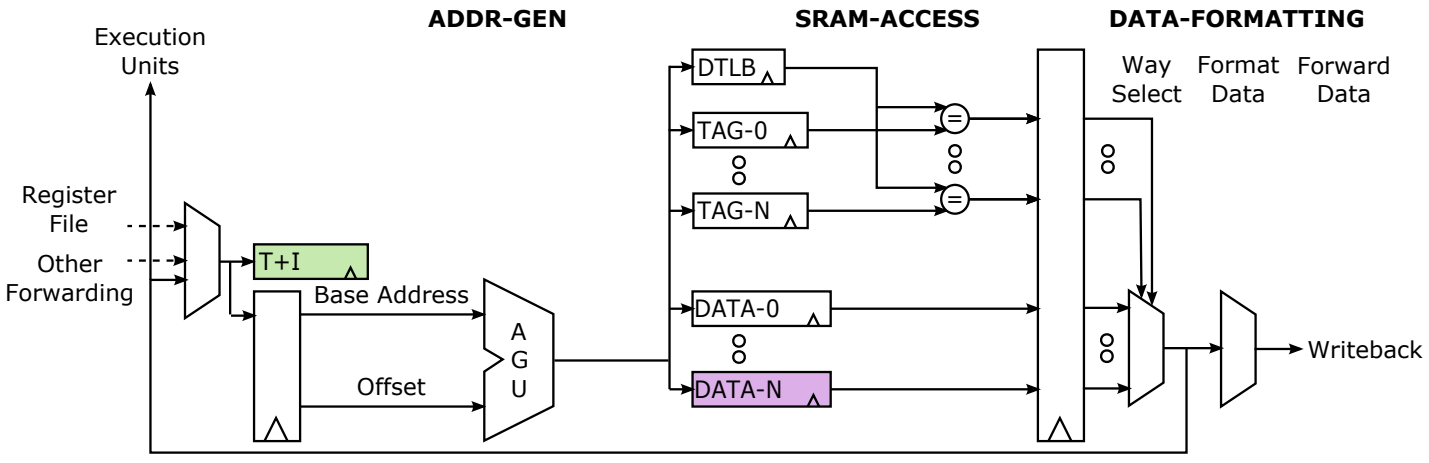
Context Aware Loads — Case4



r[2] = M[r[4]]

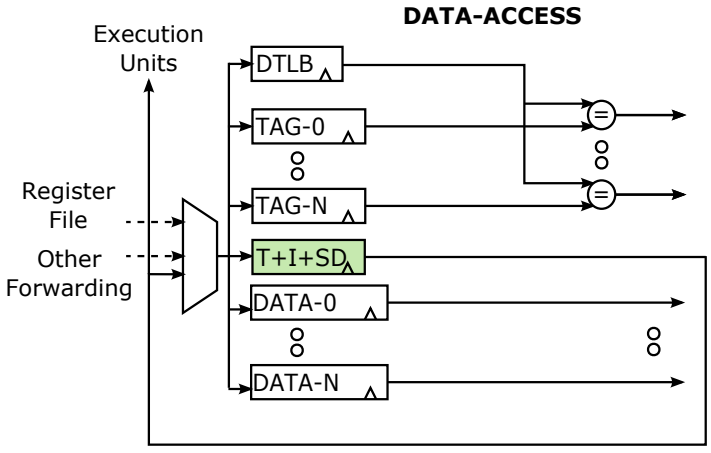
r[3] = r[3]+r[2]

Context Aware Loads — Case5



r[2] = M[r[4]+4]
r[3] = r[3]+r[2]

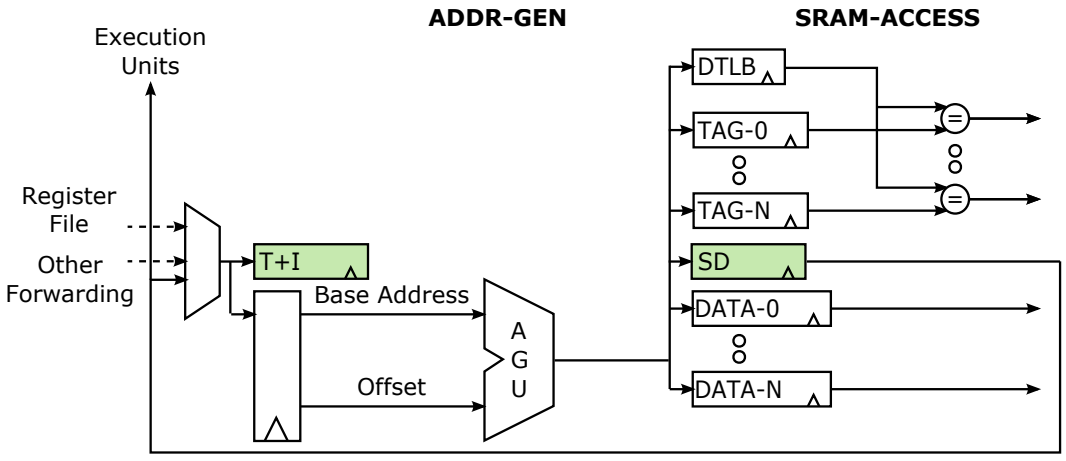
Context Aware Loads — Case6



r[2] = M[r[4]]

r[3] = r[3]+r[2]

Context Aware Loads — Case7

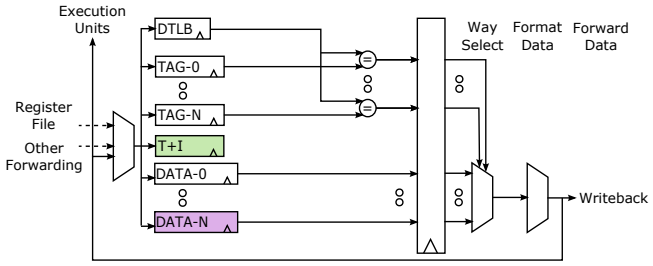


r[2] = M[r[4]+4]

r[3] = r[3]+r[2]

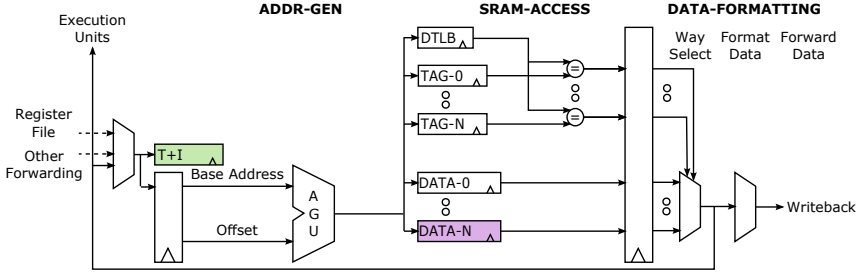
Context Aware Loads — Cases 4-7

r[2] = M[r[4]]
r[3] = r[3]+r[2]



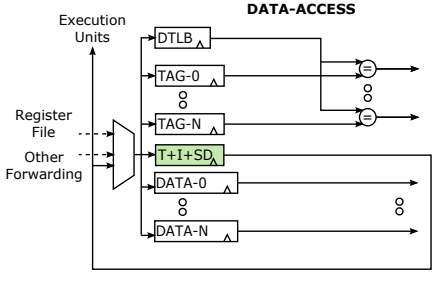
Case4:
Avoid 1 Stall
No DTLB Access
No Tag Checks
1x Data Array Access

r[2] = M[r[4]+4]
r[3] = r[3]+r[2]



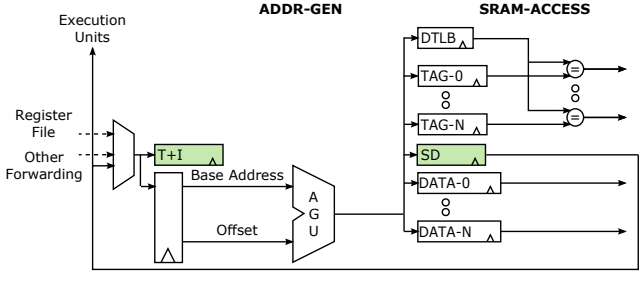
Case5:
No DTLB Access
No Tag Checks
1x Data Array Access

r[2] = M[r[4]]
r[3] = r[3]+r[2]



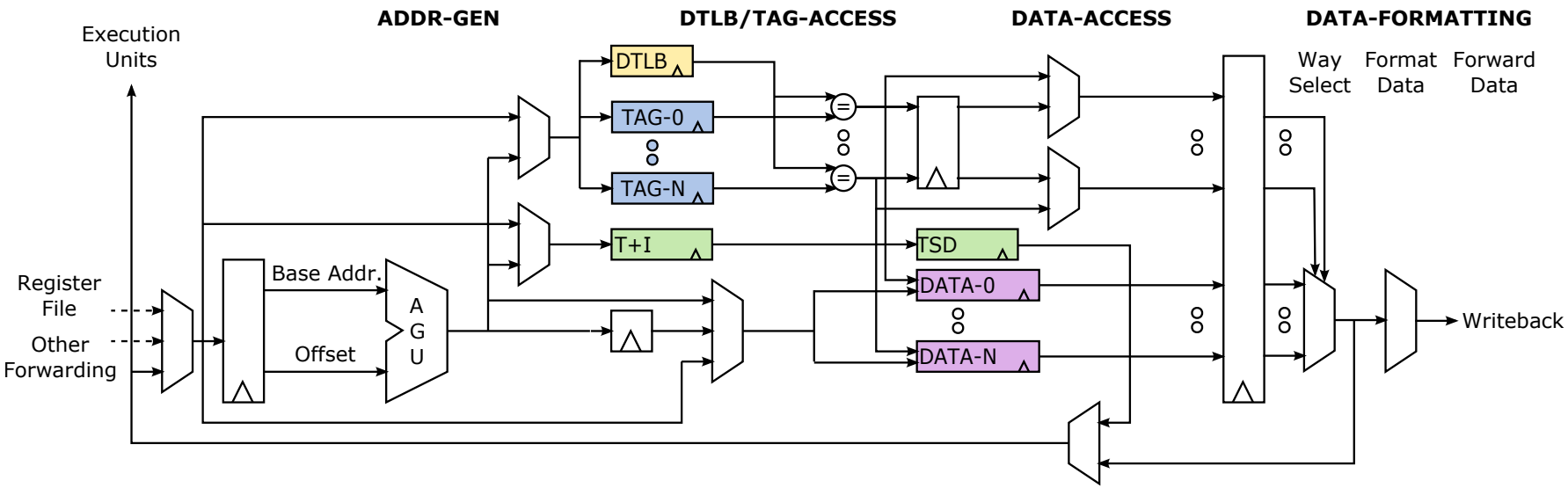
Case6:
Avoid 2 Stalls
No DTLB Access
No Tag Checks
No Data Array Access

r[2] = M[r[4]+4]
r[3] = r[3]+r[2]

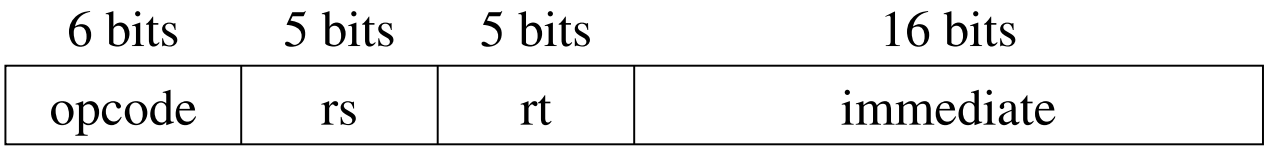


Case7:
Avoid 1 Stall
No DTLB Access
No Tag Checks
No Data Array Access

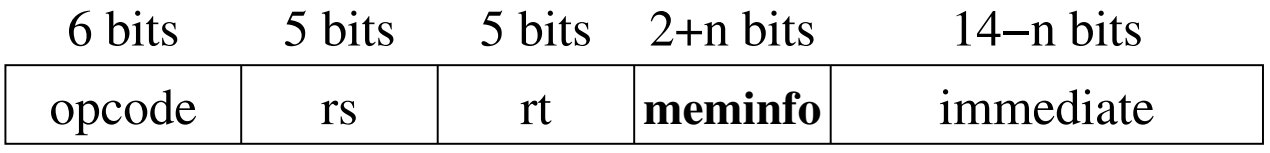
Context Aware Loads — Pipeline



Instruction Format



(a) MIPS Instruction I Format

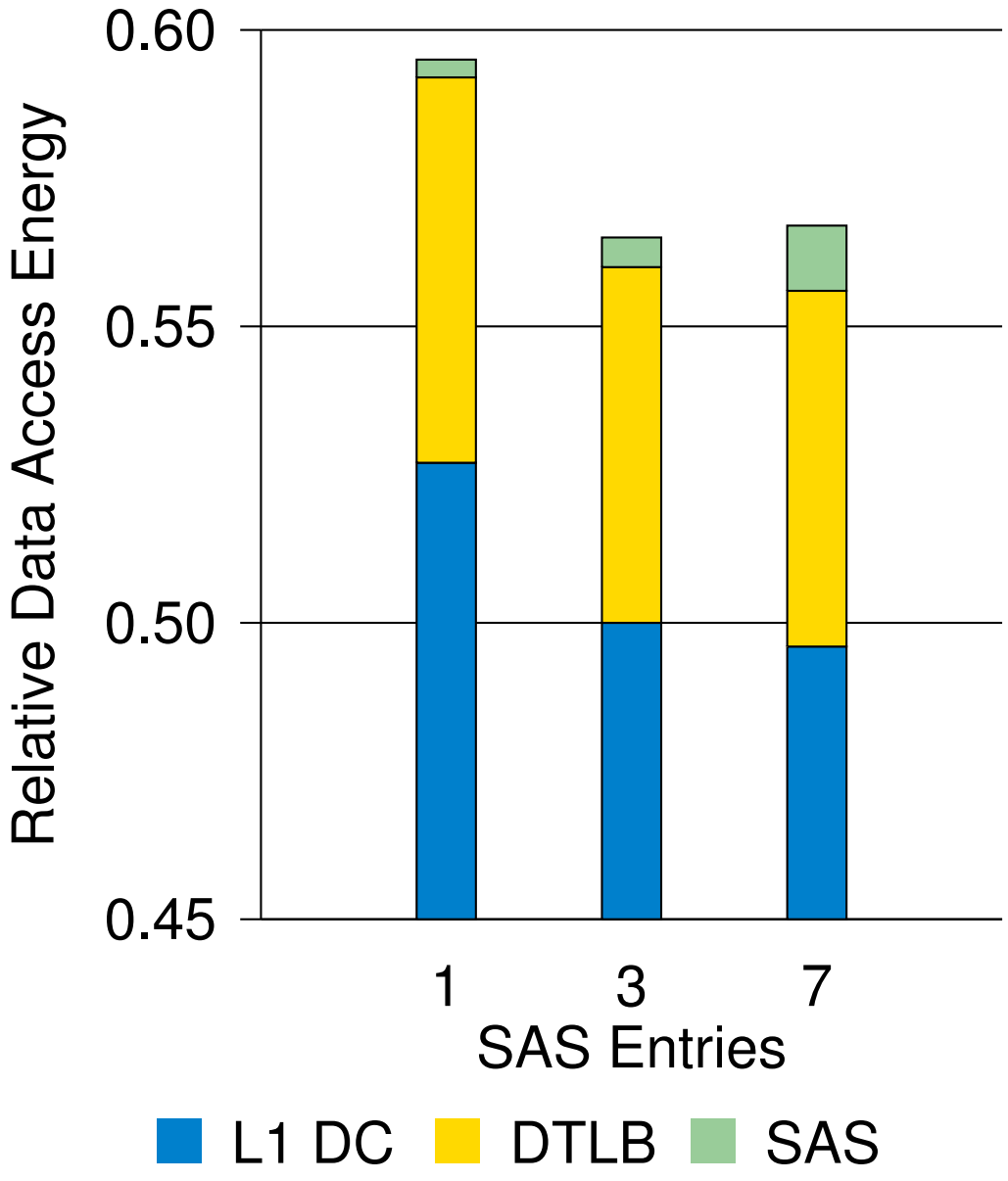


(b) Enhanced Load and Store Instruction Format

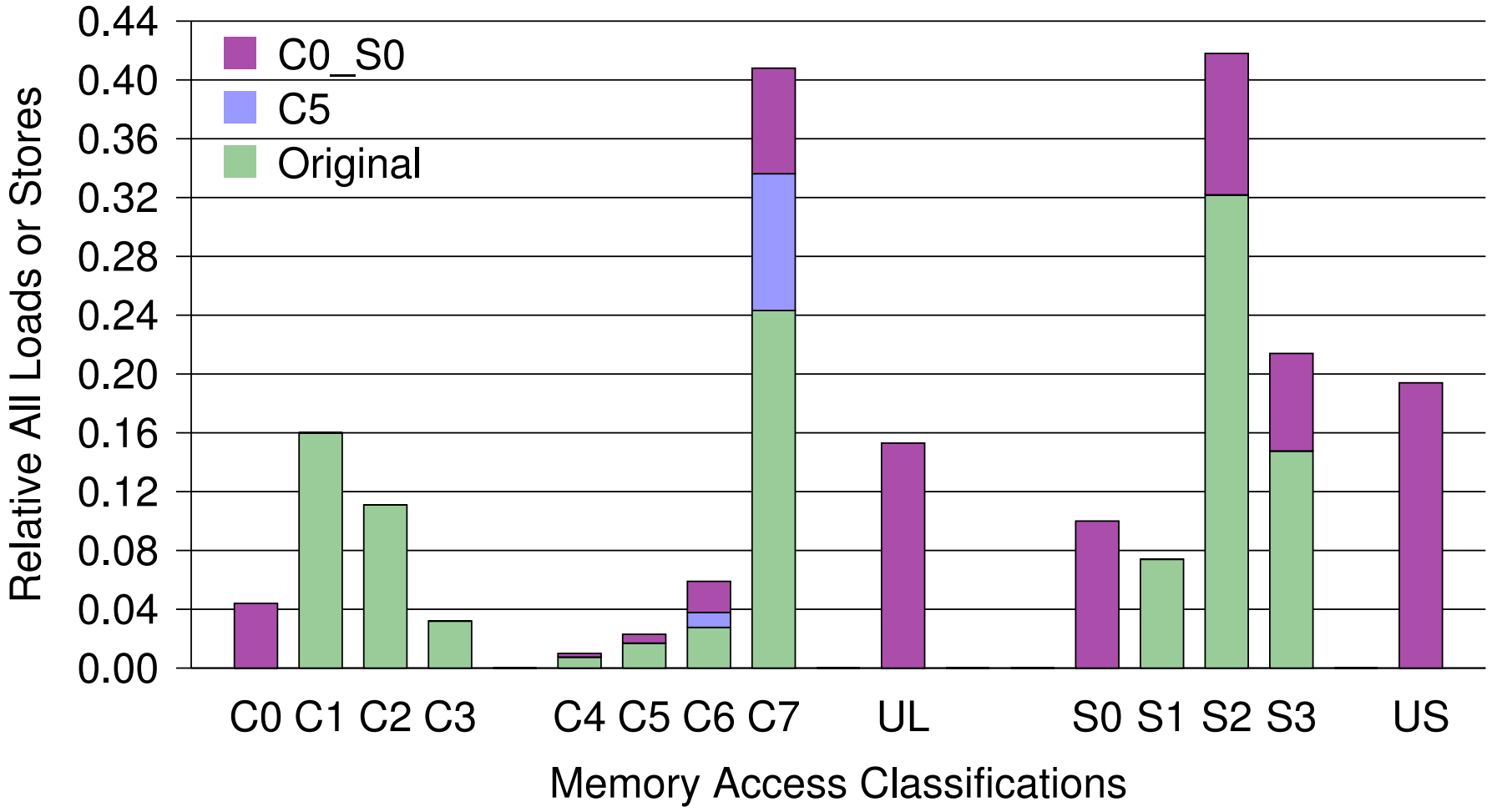
Evaluation Framework

- VPO compiler
- MiBench
- Simple Scalar
- L1 DC: 16KiB, 4-way, 32-byte line
- DTLB: 16 entries, fully associative
- Energy: P&R netlist in 65-nm technology

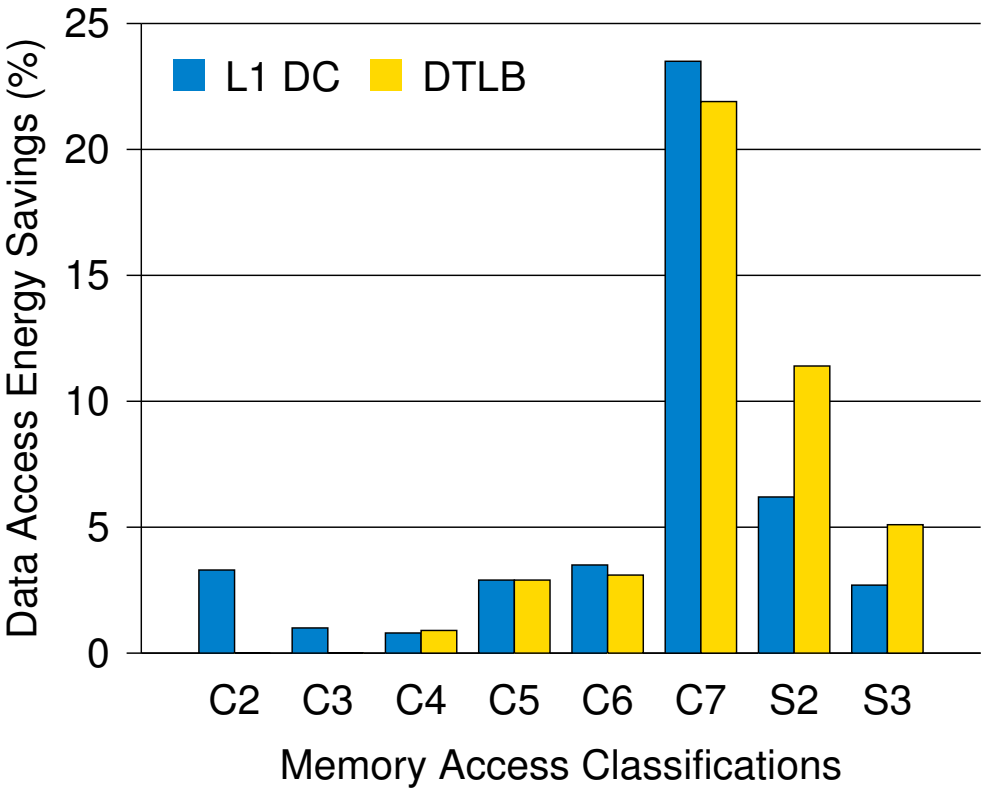
SAS Entries


0.60
0.55
0.50
0.45
Relative Data Access Energy
1
3
7
SAS Entries
L1 DC
DTLB
SAS

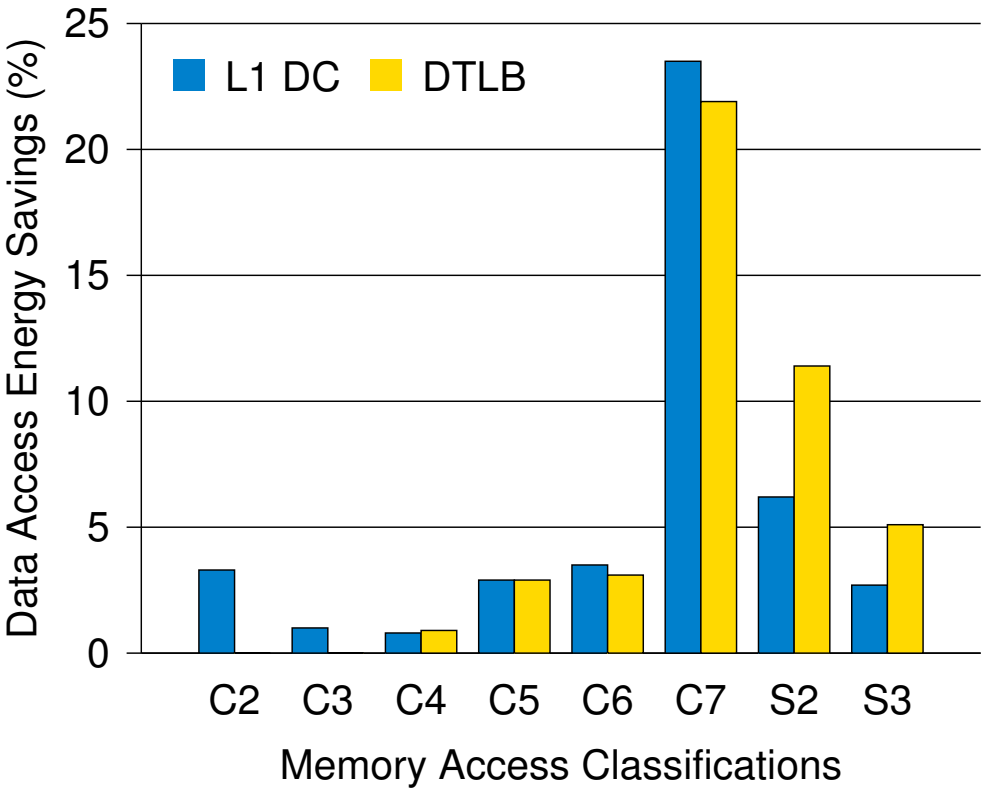
Classification of Loads and Stores

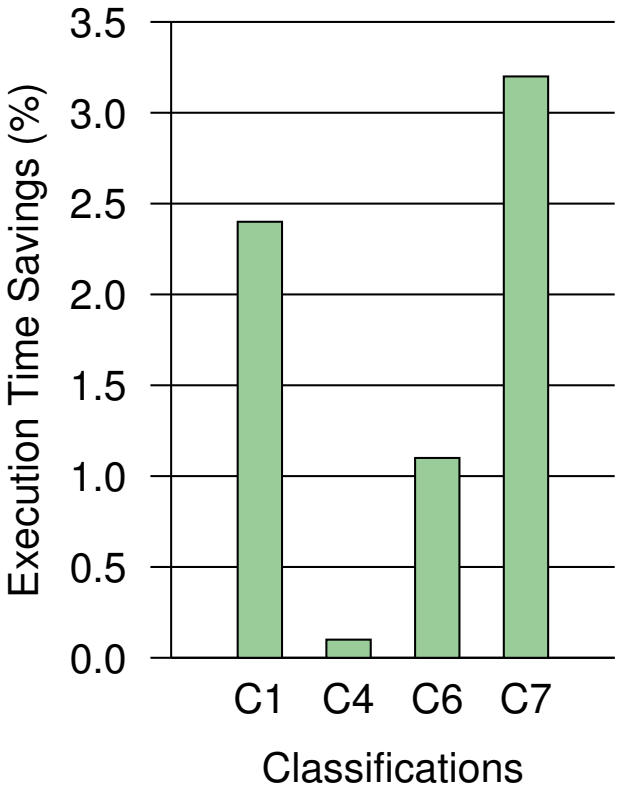


Per Case

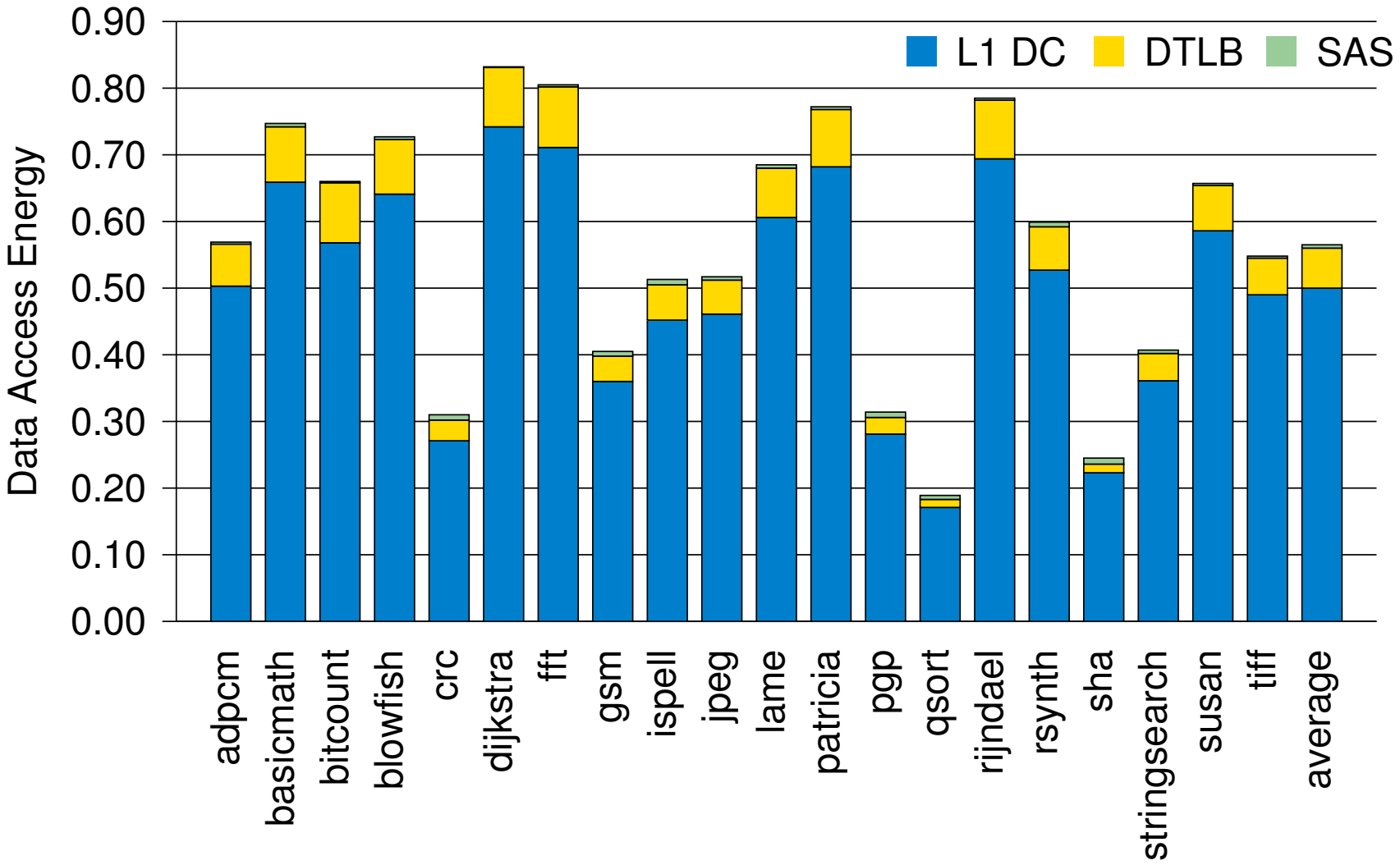

25
20
15
10
5
0
L1 DC
DTLB
Data Access Energy Savings (%)
C2
C3
C4
C5
C6
C7
S2
S3
Memory Access Classifications

Per Case

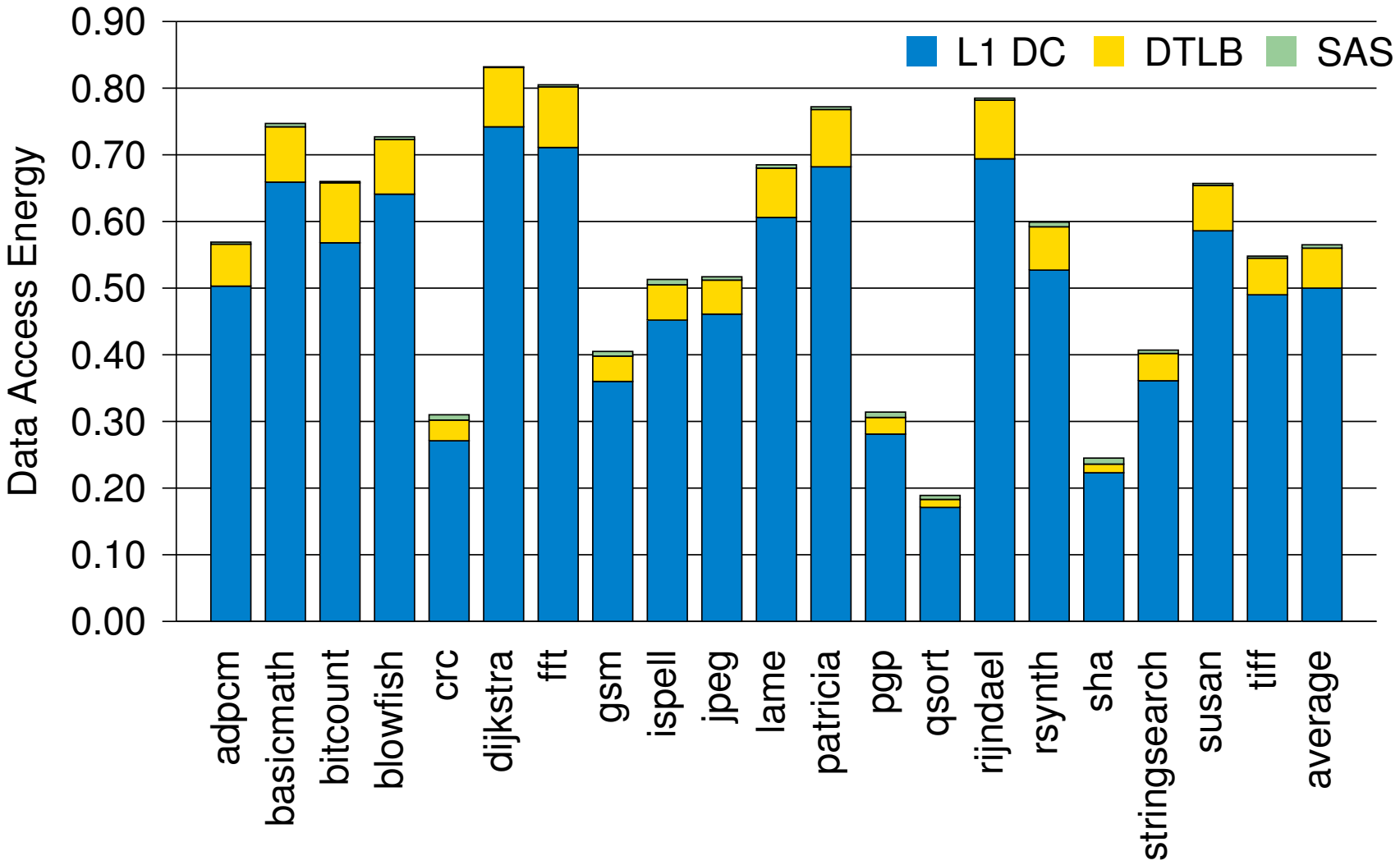

Data Access Energy Savings (%)
25
20
15
10
5
0
L1 DC
DTLB
C2
C3
C4
C5
C6
C7
S2
S3
Memory Access Classifications


Execution Time Savings (%)
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
C1
C4
C6
C7
Classifications

Energy Improvements

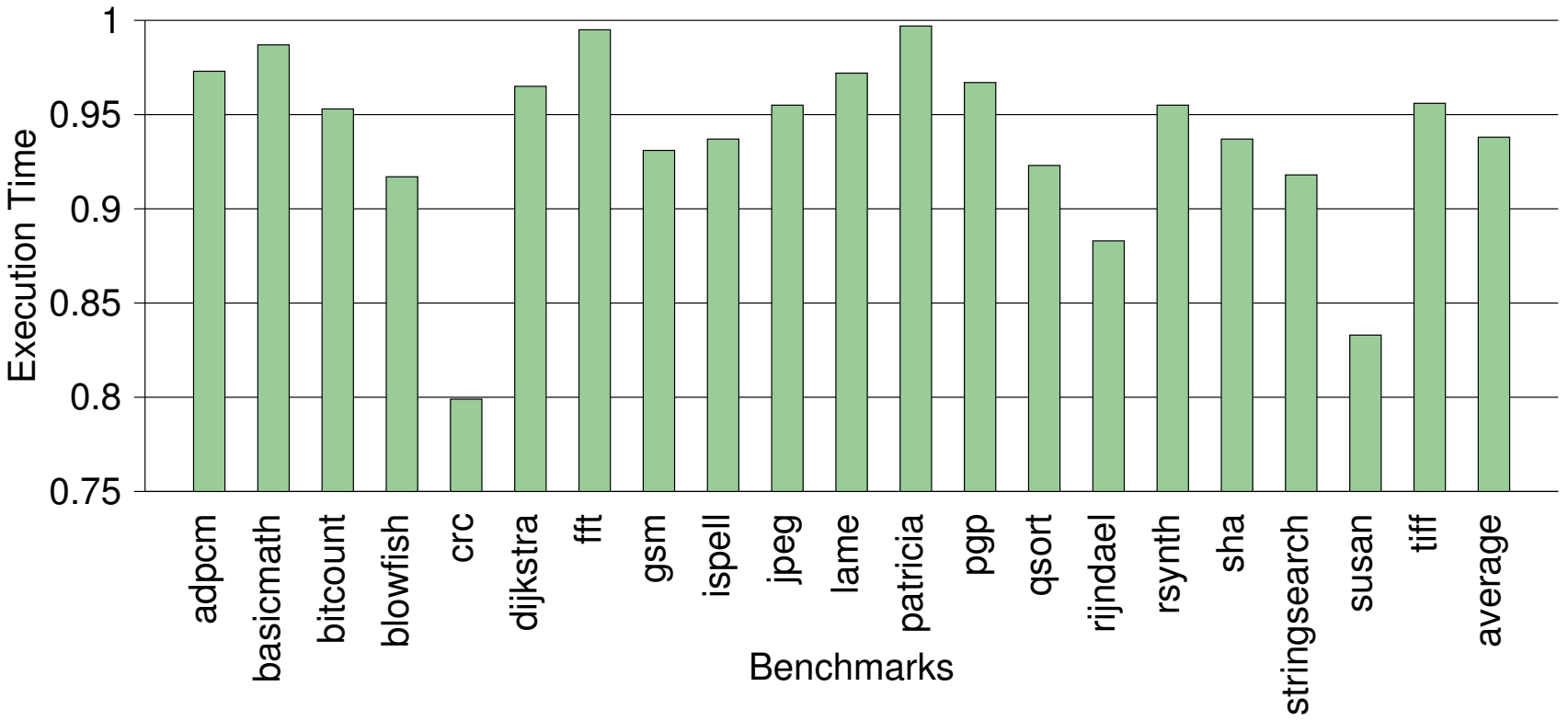


Energy Improvements

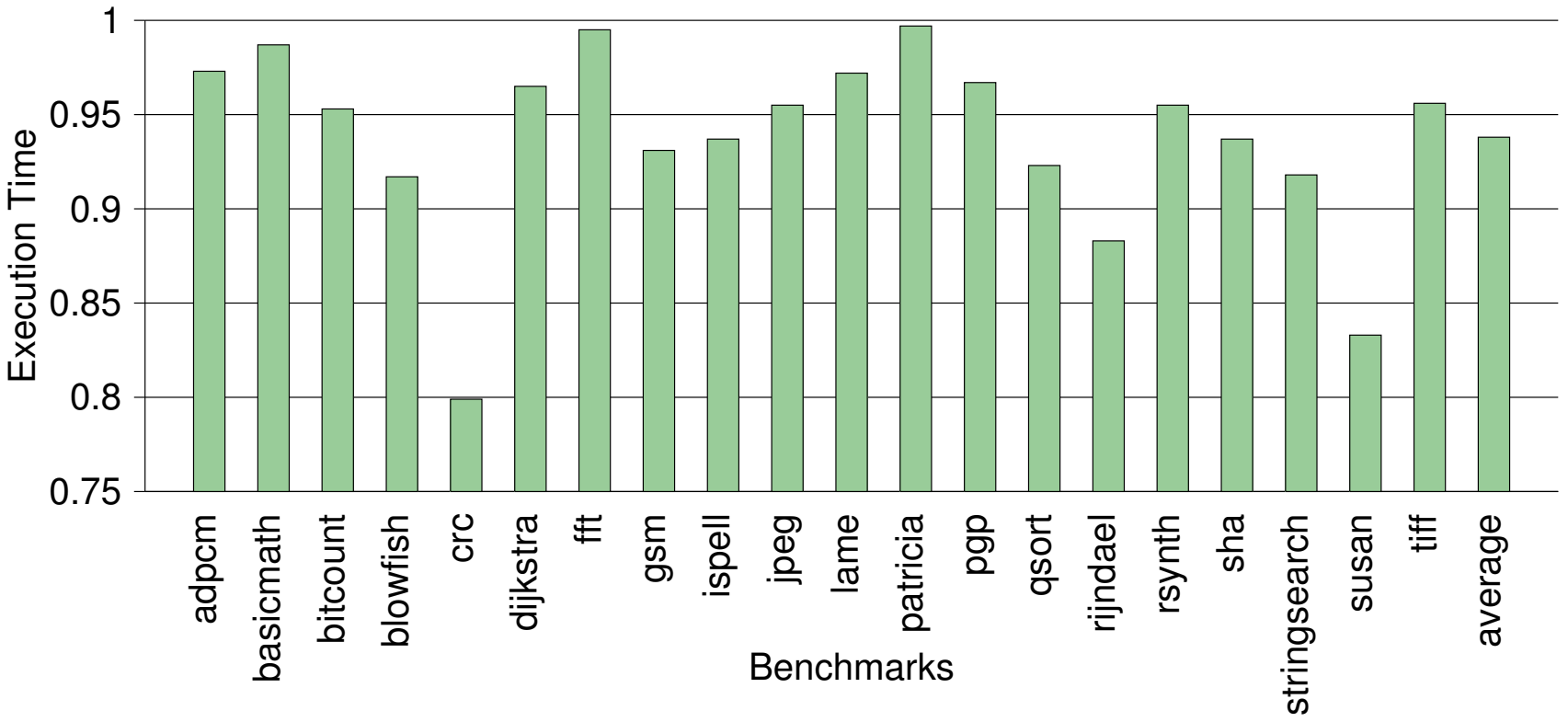


On average 43% energy usage reduction

Execution Time Improvements



Execution Time Improvements



On average 6% execution time improvement

Summary

- Conventional associative L1 caches are power hungry
- Context aware data accesses reduce L1 data cache power
- Speculative and early tag access improves performance

Summary

- Conventional associative L1 caches are power hungry
- Context aware data accesses reduce L1 data cache power
- Speculative and early tag access improves performance

- 43% reduction in L1 data cache and DTLB energy usage
- 6% performance improvement