Compiler Restructuring

Objective: restructure the sequential code (mainly loops) through data dependence analysis to identify constraints to Parallelism and to apply loop transformations that do not modify the data dependences with the idea of matching the computation with the underlying parallel machine.

1- Data dependence analysis of loops

There are 2 types of dependences or recurrences:

1) loop-independent-dependcy (LID)

no dependence from one iteration to another

2) loop-carried-dependency (LCD)

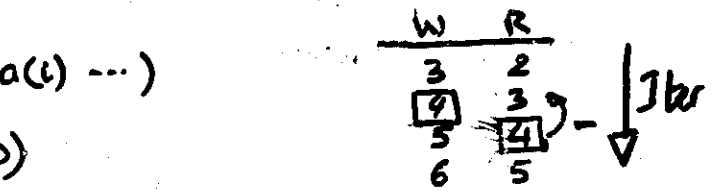
A read of an array variable is done in one iteration and a write of the same var. in another iteration

Two Types of LCD

Backward (B-LCD)

A read in one iteration and a write in a previous iteration

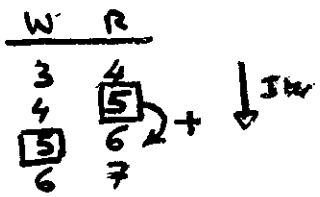
$a(i+1) = F(\cdots\ a(i)\ \cdots)$



Forward (F-LCD)

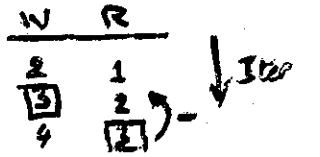
A read in one iteration and a write in a future iteration.

$a(i) = F(\cdots\ a(i+1)\ \cdots)$



Ex.

```
for i = 1, 1000
    a(i) = b(i) + a(i)
    b(i+1) = c(i) + 3
end
```



There is a B-LCD due to array b

2 - Dependence distance

A general loop for $i = l, u, s$ (s: stride)

$$a(i) = F(\cdots, a(i+K), \cdots)$$

There is an LCD if (Recurrence):

(1) dependence distance $d = \underbrace{i+K}_{\text{read}} - \underbrace{i}_{\text{write}} = K$

$K \neq 0$ and $K = K' \times s$ K' is an Integer

(2) Normalized dependence distance $d_n = \frac{d}{s} \in [1, \frac{u-l}{s}+1]$

The read is to a var. that is written in an iteration that is distant by d_n.

Ex.1 for $i = 6, 31, 3$

$$a(i) = F(\cdots a(i+6) \cdots)$$

R1: $d = i+6-i = 6 = 2 \times 3$

R2: $d_n = 6/3 = 2 \in [1, 2, \cdots, 9]$

This is an F-LCD (No prob. for pipelining)

The loop can be parallelized if array a is not shared (No coherence)

In	W	R
1	6	12
2	9	15
3	12	18
4	15	21
5	18	24
6	21	27

+2

Ex2 for $i = 4, 31, 2$

$$a(i) = F(\cdots a(i-4) \cdots)$$

R1: $d = i-4-i = -4 = -2 \times 2$

R2: $d_n = -4/2 = -2$

This is an B-LCD* (limiting Pipelining)

It cannot be parallelized over an SMP or cluster in a simple way.

In	W	R
1	4	0
2	6	2
3	8	4
4	10	6
5	12	8
6	14	10

-2

* Subjects to nbr. of stages and pipeline initiation time.

+ B-LCD with $d=K \Rightarrow \exists$ K indep. Chains that can provide K-way

Process 1		
1	4	0
3	8	4
5	12	8

B-LCD

Process 2		
2	6	2
4	10	6
6	14	10

B-LCD

Affine references

The reference is said to be affine if it is a linear function of i.

Consider $i = l, u, 1$

$\ldots = \ldots A(Ci + d) \ldots$

$A(ai+b) = \ldots\ldots\ldots$

There is a recurrence between the above references

IF: (1) $\exists\ j$ and k such that $l \le j, k \le u$

and

(2) $aj + b = Ck + d$ (this is difficult to test)

A sufficient Recurrence test is

> IF GCD (C, a) Divides d − b, then there there is a recurrence.

Ex

$\ldots = \ldots a(4i-3)$

$a(6i+1) = \ldots\ldots$

GCD (4, 6) = 2 and $\frac{1+3}{2} = 2$ ⇒ recurrence

⇒

i	6i+1	4i−3	
1	7	1	
2	[13]	5	
3	19	9	−2
4	[25]	[13]	
5	31	17	
	[37]	21	−3
	33	[25]	
	39	29	
	45	33	−4
	51	[37]	

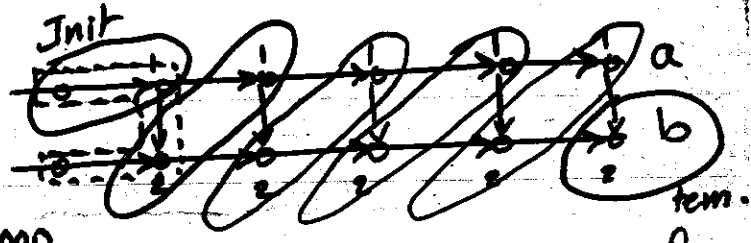
Detecting and enhancing loop-level parallelism

Ex1

```
for (i=1; i<=100; i=i+1){
    a(i+1) = a(i) + c(i);
    b(i+1) = b(i) + a(i+1);
}
```

(1) Dependence distance $d = \begin{cases} i - i-1 = -1 & \text{for array a (B-LCD)} \\ i - i-1 = -1 & \text{for array b (B-LCD)} \end{cases}$ (Read Write)

(2) A partial order from ST1 → ST2
- Cannot be interchanged (statements)
- Loop can be distributed: no data dependence loop from ST1 → ST2 → ST1 → Iteration Space



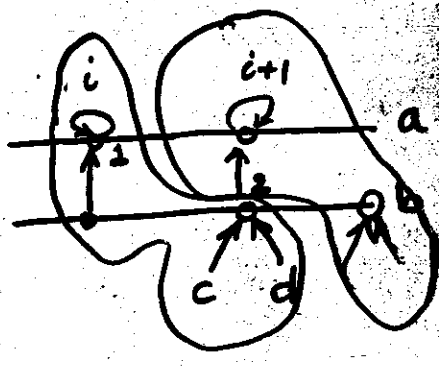
(3) To increase separation Producer → Consumer we may restructure so that each iteration contains instructions from diff. initial loop iterations.

Note: No arrows in each Red Collection (New iteration).

Ex2:

```
for (i=1; i<=100; i=i+1){
St-1 .... a(i) = a(i) + b(i);
St-2 .... b(i+1) = c(i) + d(i);
}
```

(1) Dep. Dist.: $d = \begin{cases} i - i = 0 & \text{(LID) a} \\ i - i-1 = -1 & \text{(B-LCD) b} \end{cases}$



(2) St-1 is an LID but b(i) is computed (St-2) in the previous iteration

(3) The flow of data is St-2 → St-1 without a loop.

⇒ St1 & ST2 can be interchanged

⇒ ST2 & ST1 can be distributed into 2 loops

loop (ST2 Now ST2 is an LID (Unrolling)
ST1 is an LID too. //

```
for (i=1; i<=100; i=i+1) {
st1 .... a(i) = b(i) + c(i)
st2 .... d(i) = a(i) * e(i)
```

---

for $i = 1$ to N

$$a(i+1) = F\ (\ a(i-1), \ldots\ )$$

$$d = i-1-i-1 = -2$$

⇒ Two chains (sequence of data dep. computation) : -2

(1) $3 \rightarrow 5 \rightarrow 7 \rightarrow 9$

(2) $2 \rightarrow 4 \rightarrow 6 \rightarrow 8$

iter.	Read	Write
2	1	3
3	2	4
4	3	5
5	4	6
6	5	7
7	6	8
8	7	9
9	8	10