Example Code

```
for (i=0; i<=999; i=i+1)
    x[i] = x[i] + c;
```

```
Loop:   L.D     F0,0(R1)        ;F0=array element
        ADD.D   F4,F0,F2        ;add scalar in F2
        S.D     F4,0(R1)        ;store result
        DADDUI  R1,R1,#-8       ;decrement pointer
        BNE     R1,R2,Loop      ;branch R1!=R2
```

Loop Unrolling

Unroll the loop by 4 assuming R2 is a multiple of 32

```
Loop:   L.D     F0,0(R1)
        ADD.D   F4,F0,F2
        S.D     F4,0(R1)
        DADDUI  R1,R1,#-8
        BNE     R1,R2,Loop
```

```
Loop:   L.D     F0,0(R1)
        ADD.D   F4,F0,F2
        S.D     F4,0(R1)
        L.D     F6,-8(R1)
        ADD.D   F8,F6,F2
        S.D     F8,-8(R1)
        L.D     F10,-16(R1)
        ADD.D   F12,F10,F2
        S.D     F12,-16(R1)
        L.D     F14,-24(R1)
        ADD.D   F16,F14,F2
        S.D     F16,-24(R1)
        DADDUI  R1,R1,#-32
        BNE     R1,R2,Loop
```

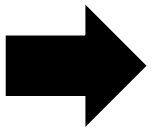
Delay

Producing result	Using result	Latency in clock cycle
FP ALU op	Another FP ALU op	3
FP ALU op	Store double	2
FP ALU op	Branch	1
Load double	FP ALU op	1

```
Loop:   L.D     F0,0(R1)
        stall
        ADD.D   F4,F0,F2
        stall
        stall
        S.D     F4,0(R1)
        DADDUI  R1,R1,#-8
        stall
        BNE     R1,R2,Loop
```

Loop Unrolling and Scheduling

```
Loop:   L.D      F0,0(R1)
        stall
        ADD.D    F4,F0,F2
        stall
        stall
        S.D      F4,0(R1)
        L.D      F6,-8(R1)
        stall
        ADD.D    F8,F6,F2
        stall
        stall
        S.D      F8,-8(R1)
        L.D      F10,-16(R1)
        stall
        ADD.D    F12,F10,F2
        stall
        stall
        S.D      F12,-16(R1)
        L.D      F14,-24(R1)
        stall
        ADD.D    F16,F14,F2
        stall
        stall
        S.D      F16,-24(R1)
        DADDUI   R1,R1,#-32
        stall
        BNE      R1,R2,Loop
```



```
Loop:   L.D      F0,0(R1)
        L.D      F6,-8(R1)
        L.D      F10,-16(R1)
        L.D      F14,-24(R1)
        ADD.D    F4,F0,F2
        ADD.D    F8,F6,F2
        ADD.D    F12,F10,F2
        ADD.D    F16,F14,F2
        S.D      F4,0(R1)
        S.D      F8,-8(R1)
        S.D      F12,-16(R1)
        S.D      F16,-24(R1)
        DADDUI   R1,R1,#-32
        stall
        BNE      R1,R2,Loop
```

Key Transformations for Loop Unrolling

1

Identify Independent Iterations

Determine that loop iterations are independent except for maintenance code

2

Register Renaming

Use different registers to avoid unnecessary constraints from name dependencies

3

Code Simplification

Eliminate extra tests and branches; adjust loop termination and iteration code

4

Memory Analysis

Determine that loads and stores from different iterations are independent by analyzing memory addresses

5

Instruction Scheduling

Schedule code while preserving dependencies needed for correct results

Limitations of Loop Unrolling

Diminishing Returns

Each additional unroll provides less overhead reduction

A decrease in the amount of overhead is amortized with each unroll

Code Size Growth

Unrolling increases code size substantially

May cause instruction cache miss rate to increase

Register Pressure

Aggressive unrolling and scheduling increases the number of live values

May create register shortage that eliminates performance gains