Handout #7
ENEE 446
Fall 2018

Loop Unrolling Ex. 1

(Parallel Loop)

```
for ( i=0; i<n; i++)
    d[i] = a[i] * b[i] + c
```

```
LOOP: L.D   F0, (R1)
      L.D   F2, (R2)
      DADDI R1, #8
      DADDI R2, #8
      MUL.D F4, F0, F2
      ADD.D F6, F4, F8
      S.D   F6, (R3)
      DADDI R3, R3, #8
      DADDI R4, R4, #1
      SLT   R5, R4, R6
      BNEZ  R5, LOOP
```

R1 - &(a[i])
R2 - &(b[i])
R3 - &(d[i])
R4 - i
R6 - n
F8 - C

Loop Unrolling Ex. 1

(Parallel Loop)

```
for (i=0; i<n; i+=2) {
    d[i] = a[i] * b[i] + C;
    d[i+1] = a[i+1] * b[i+1] + C;
}
```

```
LOOP: L.D F0, (R1)
      L.D F2, (R2)
      MUL.D F4, F0, F2
      ADD.D F6, F4, F8
      S.D F6, (R3)
     - - - - - - - - - - - - -
      L.D F0, 8(R1)
      L.D F2, 8(R2)
      MUL.D F4, F0, F2
      ADD.D F6, F4, F8
      S.D F6, 8(R3)
      DADDI R1, R1, #16
      DADDI R2, R2, #16
      DADDI R3, R3, #16
      DADDI R4, R4, #2
      SLT R5, R4, R6
      BNEZ R5, LOOP
```

Loop Unrolling Ex 1

(Parallel Loop)

```
LOOP:  L.D F0, (R1)
       L.D F2, (R2)
       L.D F10, 8(R1)
       L.D F12, 8(R2)
       DADDI, R1, R1, #16
       DADDI R2, R2, #16
       MUL.D F4, F0, F2
       MUL.D F14, F10, F12
       ADD.D F6, F4, F8
       ADD.D F16, F14, F8
       S.D F6, (R3)
       S.D F16, 8(R3)
       DADDI R3, R3, #16
       DADDI R4, R4, #2
       SLT R5, R4, R6
       BNEZ R5, LOOP
```

Loop Unrolling Ex 2
(Sequential Loop)

```
max = a[0];
for (i=1; i<n; i++)
   if (a[i] > max)
      max = a[i];
```

R2 - &a[i]
R4 - i
R8 - max
R9 - n

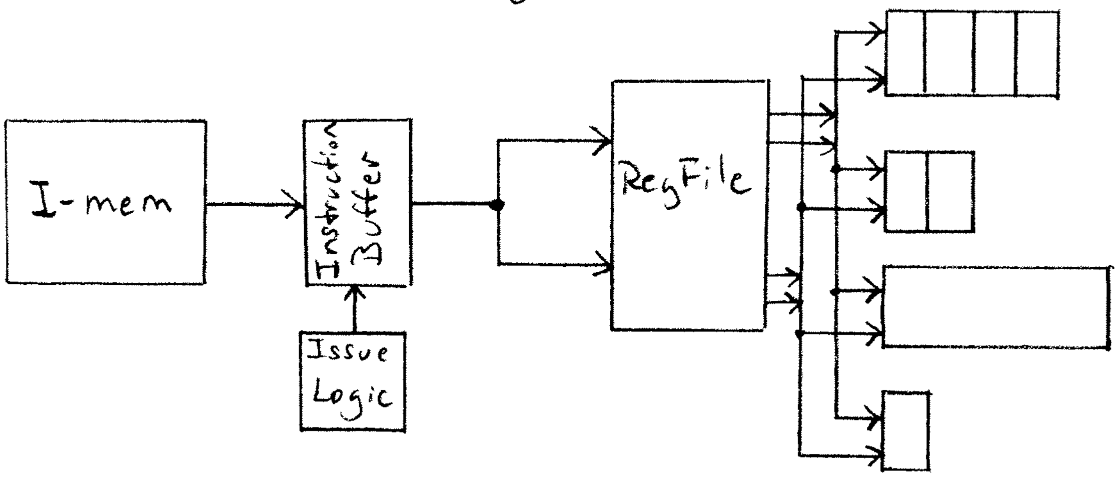
```
LOOP:   LD   R1, (R2)
        SGT  R3, R1, R8
        BEQZ R3, NOMAX
        DADD R8, R1, R0
NOMAX:  DADDI R2, R2, #4
        DADDI R4, R4, #1
        SLT  R5, R4, R9
        BNEZ R5, LOOP
```

Loop Unrolling Ex. 2

(Sequential Loop)

```
LOOP:   LD   R1, (R2)
        SGT  R3, R1, R8
        BEQZ R3, NOMAX1
        DADD R8, R1, R0
NOMAX1: LD   R1, 4(R2)
        SGT  R3, R1, R8
        BEQZ R3, NOMAX2
        DADD R8, R1, R0
NOMAX2: DADDI R2, R2, #8
        DADDI R4, R4, #2
        SLT  R5, R4, R9
        BNEZ R5, LOOP
```

Instruction Schedule for Dynamic Issue (in-order)



Ex: Issue Rules - 2 per cycle, at most 1 LD/SD, at most 1 float
Latency: LD - 2, int - 1, F* - 3, F+ - 3

```
LOOP:  L.D    F0, (R1)
       L.D    F2, (R2)
       L.D    F4, 8(R1)
       L.D    F6, 8(R2)
       DADDI  R1, R1, #16
       DADDI  R2, R2, #16
       MUL.D  F8, F0, F2
       MUL.D  F10, F4, F6
       ADD.D  F12, F8, F16
       ADD.D  F14, F10, F16
       S.D    F12, (R3)
       S.D    F14, 8(R3)
       DADDI  R3, R3, #16
       DADDI  R4, R4, #2
       SLT    R5, R4, R6
       BNEZ   R5, LOOP
```

1	L.D F0, (R1)	-
2	L.D F2, (R2)	-
3	L.D F4, 8(R1)	-
4	L.D F6, 8(R2)	DADDI R1, R1, #16
5	DADDI R2, R2, #16	MUL.D F8, F0, F2
6	MUL.D F10, F4, F6	-
7	-	-
8	ADD.D F12, F8, F16	-
9	ADD.D F14, F10, F16	-
10	-	-
11	S.D F12, (R3)	-
12	S.D F14, 8(R3)	DADDI R3, R3, #16
13	DADDI R4, R4, #2	
14	SLT R5, R4, R6	
15	BNEZ R5, LOOP	

16 instructions ⇒ 15 cycles

Instruction Schedule for Dynamic Issue (in-order)

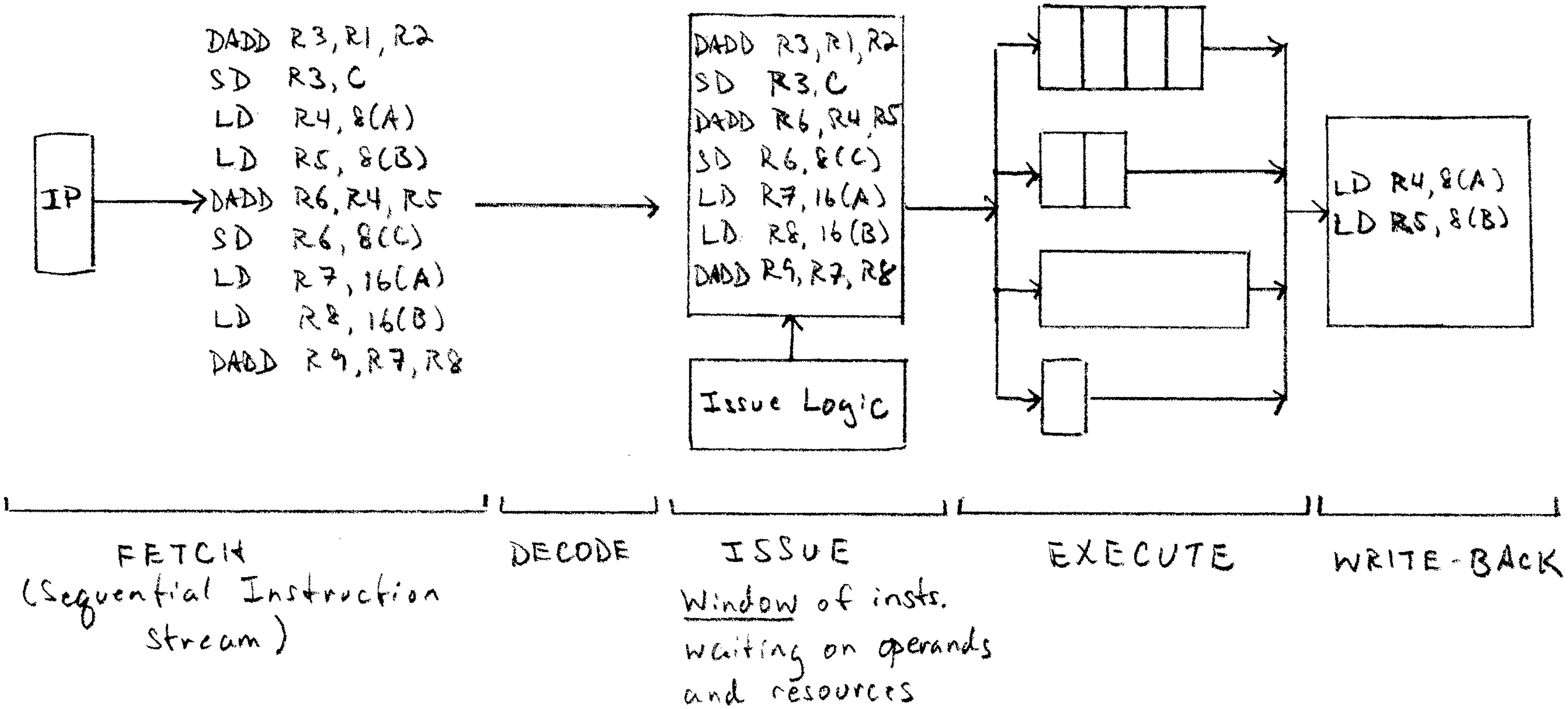
Ex: Issue Rules - 2 per cycle, at most 1 LD/SD, at most 1 float
Latency: LD - 1, int - 0, F* - 2, F+ - 2

```
LOOP: L.D F0, (R1)
      DADDI R1, R1, #16
      L.D F2, (R2)
      DADDI R2, R2, #16
      L.D F4, -8(R1)
      DADDI R4, R4, #2
      L.D F6, -8(R2)
      MUL.D F8, F0, F2
      DADDI R3, R3, #16
      MUL.D F10, F4, F6
      SLT R5, R4, R6
      ADD.D F12, F8, F16
      ADD.D F14, F10, F16
      S.D F12, -16(R3)
      S.D F14, -8(R3)
      BNEZ R5, LOOP
```

1	L.D F0, (R1)	DADDI R1, R1, #16
2	L.D F2, (R2)	DADDI R2, R2, #16
3	L.D F4, -8(R1)	DADDI R4, R4, #2
4	L.D F6, -8(R2)	MUL.D F8, F0, F2
5	DADDI R3, R3, #16	-
6	SLT R5, R4, R6	MUL.D F10, F4, F6
7	ADD.D F12, F8, F16	-
8	-	-
9	ADD.D F14, F10, F16	-
10	S.D F12, -16(R3)	-
11	-	-
12	S.D F14, -8(R3)	BNEZ R5, LOOP

16 instructions ⇒ 12 cycles

Dynamic Instruction Scheduling: Basic Idea



Static Instruction Scheduling

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
iteration i	① L.D F0, A(R1)	F	D	I	E	M	W																								
	② L.D F2, B(R1)		F	D	I	E	M	W																							
	③ MUL.D F4, F0, F2			F	D	I	-	E	E	E	E	W																			
	④ ADD.D F6, F4, F8				F	D	-	-	-	-	I	E	E	E	E	W															
	⑤ S.D F6, D(R1)					F	-	-	-	-	D	-	-	I	E	M	W														
	⑥ DADDI R1, R1, #8										F	-	-	D	I	E	M	W													
	⑦ SLT R3, R1, R2													F	D	I	E	M	W												
	⑧ BNEZ R3, LOOP														F	D	I	E	M	W											
iteration i+1	⑨ L.D F0, A(R1)															-	-	-	F	D	I	E	M	W							
	⑩ L.D F2, B(R1)																			F	D	I	E	M	W						
	⑪ MUL.D F4, F0, F2																				F	D	I	-	E	E	E	E	W		
	⑫ ADD.D F6, F4, F8																					F	D	-	-	-	-	I	E	E	E
	⑬ S.D F6, D(R1)																						F	-	-	-	-	D	-	-	I
	⑭ DADDI R1, R1, #8																								⋮						
	⑮ SLT R3, R1, R2																														
	⑯ BNEZ R3, LOOP																														

```
for (i=0; i<n; i++) {
    D[i] = A[i] · B[i] + C;
}
```

i → R1, C → F8, n·8 → R2

MUL.D - 3 cycles
ADD.D - 3 cycles

Dynamic Instruction Scheduling

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
iteration i	① L.D F0, A(R1)	F	D	I	E	M	W																								
	② L.D F2, B(R1)		F	D	I	E	M	W																							
	③ MUL.D F4, F0, F2			F	D	I	-	E	E	E	E	W																			
	④ ADD.D F6, F4, F8				F	D	I	-	-	-	-	E	E	E	E	W															
	⑤ S.D F6, D(R1)					F	D	I	-	-	-	-	-	-	E	M	W														
	⑥ DADDI R1, R1, #8	complete overlap					F	D	I	E	M	W																			
	⑦ SLT R3, R1, R2							F	D	I	E	M	W																		
	⑧ BNEZ R3, LOOP								F	D	I	E	M	W																	
iteration i+1	⑨ L.D F0, A(R1)									-	-	-	F	D	I	E	M	W													
	⑩ L.D F2, B(R1)													F	D	I	E	M	W												
	⑪ MUL.D F4, F0, F2														F	D	I	-	E	E	E	E	W								
	⑫ ADD.D F6, F4, F8															F	D	I	-	-	-	-	E	E	E	E	W				
	⑬ S.D F6, D(R1)																F	D	I	-	-	-	-	-	-	E	M	W			
	⑭ DADDI R1, R1, #8																	F	D	I	E	M	W								
	⑮ SLT R3, R1, R2																		F	D	I	E	M	W							
	⑯ BNEZ R3, LOOP																			F	D	I	E	M	W						

```
for (i=0; i<n; i++) {
    D[i] = A[i] · B[i] + c;
}
```

i → R1, c → F8, n·8 → R2

MUL.D - 3 cycles
ADD.D - 3 cycles