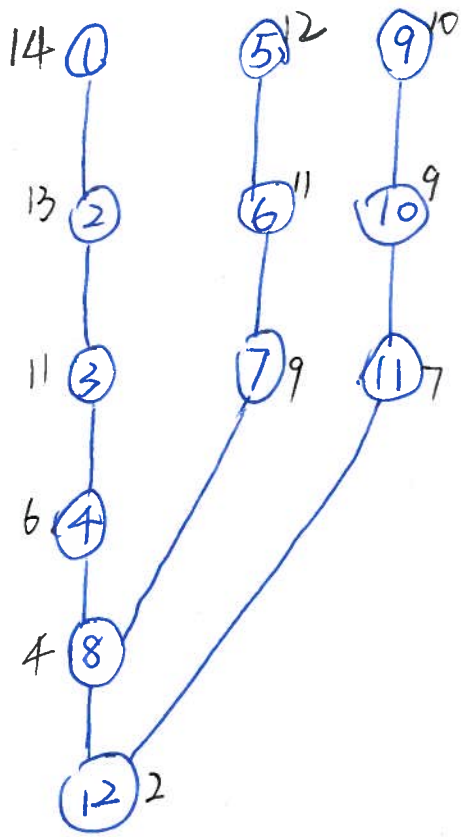
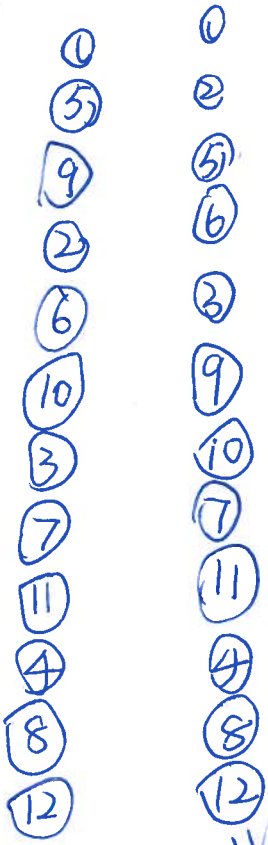
1. DDD with scheduling priority



b. Scheduling



original

	INT1	INT2	MEM
0	1		
1	2		
2			
3			3
4			
5			
6			
7			
8	4	5	
9	6		
10			7
11			
12			
13			
14			
15			
16	8	9	
17	10		
18			
19			11
20			
21			
22			
23			
24	12		

Forward:

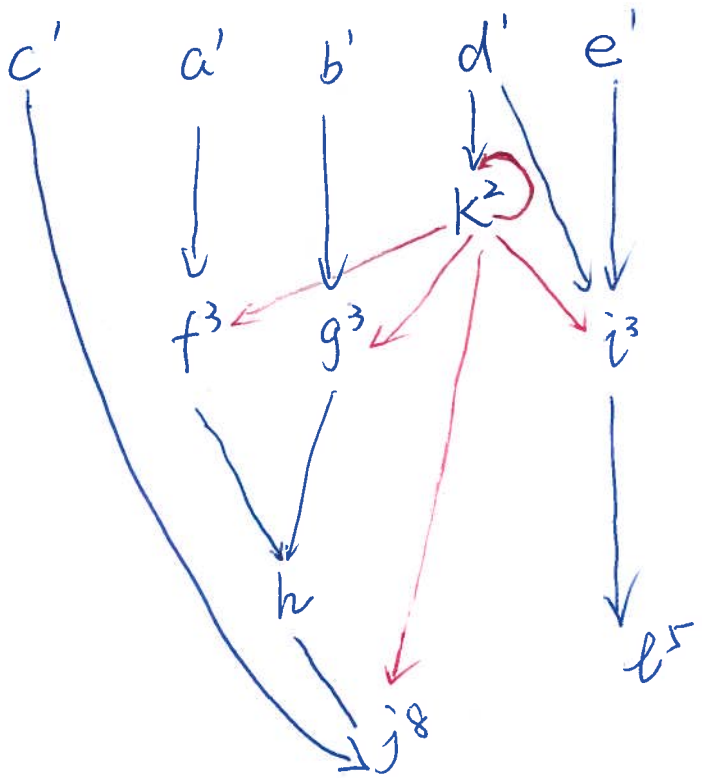
	INT1	INT2	MEM
0	1	5	
1	9		
2	6	10	
3			3
4			7
5			11
6			
7			
8	4		
9			
10	8		
11			
12	12		

Since 12 < 24
It's optimal.

Backward:

	INT1	INT2	MEM
0	1		
1	2		
2	5		
3	6		3
4	9		
5	10		7
6			
7			11
8	4		
9			
10	8		
11	12		

2.



RC

$\lceil 7/2 \rceil = 4$

for load and store

$\lceil 3/1 \rceil = 3$

So $RC = 4$

$DC = 1/1 = 1$

So $minII = 4$

Since K is antidependence with f g i j. It need to wait ~~for~~ them all done.

	Func 1	Func 2
K1	f	i
K2	g	
K3		
K4		
K5	h	
K6		
K7	j	k
K8	l	

since all include in loop. no necessary to generate prologue and epilogue.

a addI r_{arp} @x ⇒ $r_{@x}$

b addI r_{arp} @y ⇒ $r_{@y}$

c addI r_{arp} @z ⇒ $r_{@z}$

d loadI 0 ⇒ r_{ctr}

e loadI 792 ⇒ r_{ub}

f L1 = loadA$_0$ r_{ctr}, $r_{@x}$ ⇒ r_x

g loadA$_0$ r_{ctr} $r_{@y}$ ⇒ r_y

h mult r_x, r_y ⇒ r_z

i cmp_LT r_{ctr} r_{ub} ⇒ r_{cc}

j store A$_0$ r_z ⇒ r_{ctr} $r_{@i}$

k addI r_{ctr} 4 ⇒ r_{ctr}

l cbr r_{cc} → L1 L2

m L2 ...

but if can change j. store A0 $r_2 \Rightarrow r_{ctr}, r_{@z}$

to store A0 $r_2 \Rightarrow r_{ctr-4}, r_{@z}$, the kernel scheduling would be:

	Fun1	Fun2
P_1	f	
P_2	g	i
P_3		K addI $r_{ctr,4} \Rightarrow r_{ctr}$
P_4	l. if $r_{ctr} \geq r_{ub}$ goto e1	
K_1	f	h
K_2	g	i
K_3	j $r_2 \Rightarrow r_{ctr-4}, r_{@z}$	K
K_4	l if $r_{ctr} \geq r_{ub}$ goto e1	nop
e_1	nop	h
e_2	nop	nop
e_3	j $r_2 \Rightarrow r_{ctr-4}, r_{@z}$	nop
e_4	goto L2	

3.

1. loadAI rarp -4 ⇒ r_3 ~~VR1~~

	r_1	r_2	r_3
name	⊥	⊥	~~⊥~~ VR1
next	∞	∞	~~∞~~ 5
Free	T	T	~~T~~ F

~~r3~~ r2 r1

2. loadAI rarp -8 ⇒ r_2 ~~VR2~~

	r_1	r_2	r_3
name	⊥	~~⊥~~ VR2	VR1
next	∞	~~∞~~ 4	5
Free	T	~~T~~ F	F

~~r2~~ r1

3 loadI 4 ⇒ r_1 ~~VR3~~

	r_1	r_2	r_3
name	~~⊥~~ VR3	VR2	VR1
next	~~∞~~ 4	4	5
Free	~~T~~ F	F	F

~~r1~~

4 mult ~~VR2~~, VR3 ⇒ r_1 ~~VR4~~

	r_1	r_2	r_3
name	~~VR3~~ VR4	VR2	VR1
next	~~5 10 ∞~~ 9	9	5
Free	~~T~~ F	F	F

r1

Spill store r1 ⇒ rarp. 16

5. add r_3 ~~VR1~~, r_1 ~~VR4~~ ⇒ r_1 ~~VR5~~

	r_1	r_2	r_3
name	~~VR4~~ ~~⊥~~ ~~VR5~~ ⊥	VR2	~~VR1~~ ⊥
next	5 ~~∞ ∞~~	9	~~5~~ ∞
free	F ~~T~~ ~~T~~ F	F	~~F~~ T

r1 ~~r2~~ r3

6. loadi 10 ⇒ r_1 ~~VR6~~

	r_1	r_2	r_3
name	~~⊥~~ VR6	VR2	⊥
next	~~∞~~ 7	9	∞
free	~~T~~ F	F	T

~~r2~~ r3

7. store r_1 ~~VR6~~ ⇒ r_3 ~~VR5~~

	r_1	r_2	r_3
name	~~VR6~~ ⊥	VR2	~~⊥~~ ~~VR5~~ ⊥
next	~~T ∞~~	9	~~∞~~ ∞
free	~~F~~ T	F	~~T~~ ~~F~~ T

r3 ~~r2~~ ~~r1~~

8. loadAI rarp. -12 ⇒ r_3 ~~VR7~~

	r_1	r_2	r_3
name	⊥	VR2	~~⊥~~ VR7
next	∞	9	~~∞~~ 9
free	T	F	~~T~~ F

~~r3~~ r1

9. add r_3 r_2 r_2
vr7 vr2 ⇒ vr8

	r_1	r_2	r_3
name	⊥	vr2 ⊥ vr8	vr7 ⊥
next	∞	9 10	9 ∞
free	T	F F	F T

~~r_2~~
r_3
r_1

10. .
spill first.

loadAI rarp, -16 ⇒ r_3
vr3

	r_1	r_2	r_3
name	⊥	vr8	⊥ vr_3
next	∞	10	∞ 11
free	T	F	F F

~~r_2~~
r_1

mult r_2 r_3 r_2
vr8 . vr3 ⇒ vr9

	r_1	r_2	r_3
name	⊥	vr8 ⊥ vr9	vr3
next	∞	∞ 11	11
free	T	F F	F

~~r_2~~
r_1

11. store r_3 r_2
vr3 ⇒ vr9

	r_1	r_2	r_3
name	⊥	vr9 ⊥	vr3 ⊥
next	∞	∞	∞
free	T	F T	F T

r_2
r_3
r_1

Done.

4. a) live range

interference graph

a : (1-8, 9-10)

b : (2 ~ 10)

d : (5 ~ 6, 8 ~ 9)

x : (3 ~ 4, 6, 7 ~ 9)

b) spill cost

a 7/3

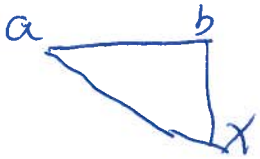
b 4/3

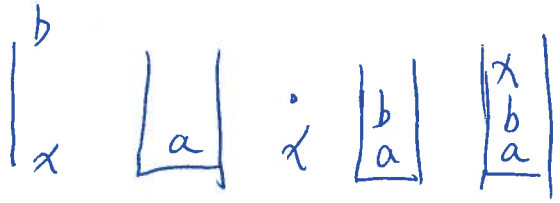
d 4/3

x 6/3

b = d < x < a

spill d = r4



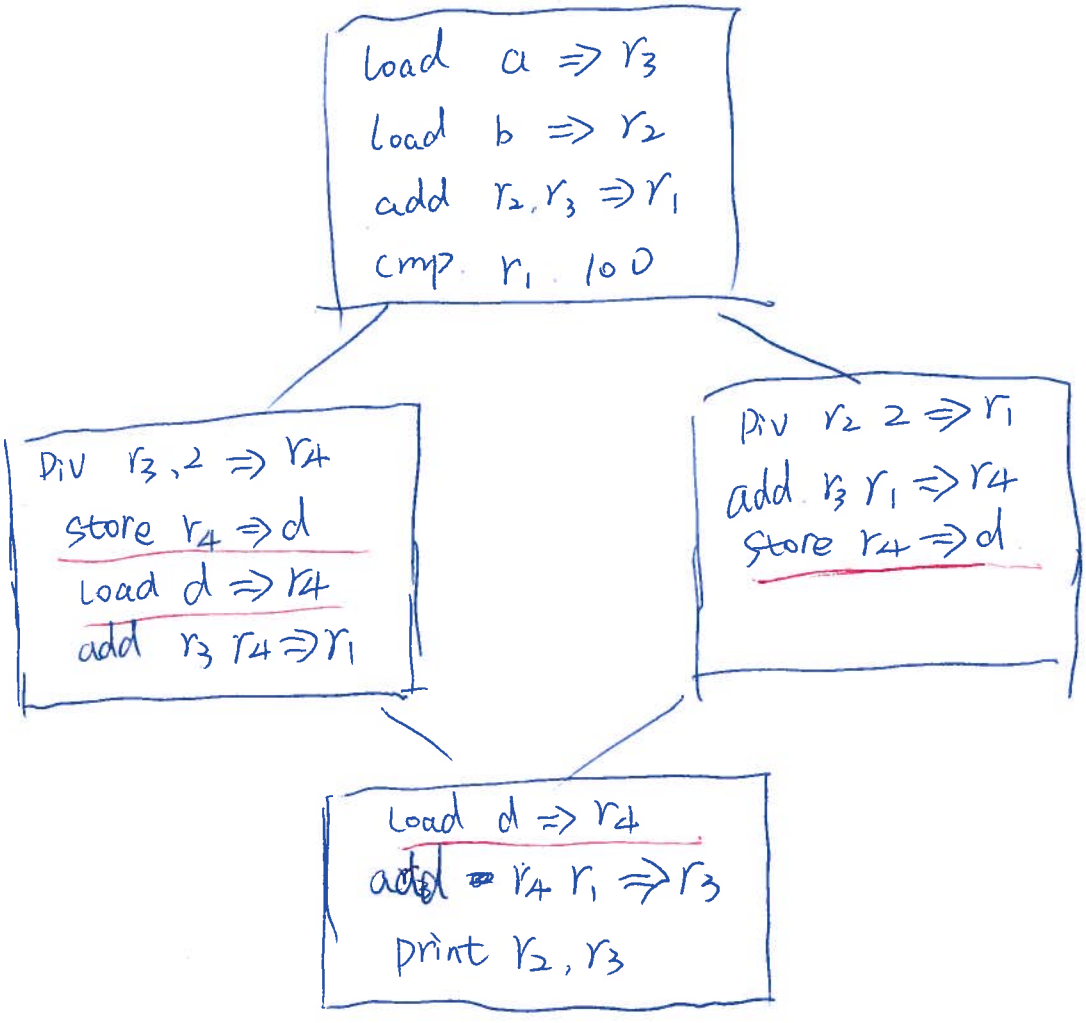


x ⇒ r1 | b / a |

x ⇒ r1, b ⇒ r2 | a |

x ⇒ r1, b ⇒ r2, a ⇒ r3 | |

c.



d.

~~top~~ down live range

a = {B1 B2 B3 B4}

b = {B1 B3 B4}

d = {B2 B3 B4}

x = {B1 B2 B3 B4}