Problem 1.1, 1.2, 1.3, 1.4

- 0x00c6ba23=0b0000 0000 1100 0110 1011 1010 0010 0011
- Opcode= 010 0011 (S-type instruction)
- Imm[4:0]=10100 Imm[11:5]=0000 000
  Imm[11:0] = 0000 0001 0100 = 20
- Rs2= 01100=12 Rs1=01101=13
- Func=011 (sd instruction)
- "sd x12, 20(x13)"

- ALU performs "add" function
- PC+4= 0x00c6ba23+4= 0x00c6ba27
- RegWrite=0 (no write back) , ALUSrc=1 (from sign extension), Branch=0 (not a branch), MemWrite=1, MemRead=0 (write to mem), MemtoReg=X (don't case, since no write back)
- ALU input1=Reg[x13], input2=20

Problem 2

- Assume
  - Latency to read PC for each instruction
  - Register file read and write latency are both included in 150ps.
  - ALUSrc Mux uses the earliest input available
- 2.1 R-type:
  PC read+IM+RF read/write+MUX+ALU+MUX
  70 + 250 + 170 + 25 + 200 + 25 = 740ps
- 2.2 ld:
  PC read+IM+RF read/write+ALU+DM+MUX
  70 + 250 + 170 + 200 + 250 + 25 = 965 ps
- 2.3 sd:
  PC read+IM+RF read/write+ALU+DM
  70 + 250 + 170 + 200 + 250 = 940
- 2.4 beq:
  PC read+IM+RF read/write+MUX+ALU+single gate+MUX
  70 + 250 + 170 + 25 + 200 + 5 + 25 = 745
- 2.5 I-type:
  PC read+IM+RF read/write+MUX+ALU+MUX
  70 + 250 + 170 + 25 + 200 + 25 = 740ps

Problem 3-1 (Assume asynchronous operation for each type)

- Original processor has the average time per instruction is .52*740 + .25*965 + .11*940 + .12 * 745 = 818.85ps
- New processor has the average time per instruction is .52*750 + .25*(1-0.12)*975 + .11 *(1-0.12)*950 + .12 * 755 = 786.46ps

- Thus, the speedup would be 818.85/786.46 =1.04

Problem 3-1 (Assume synchronous operation for each type)

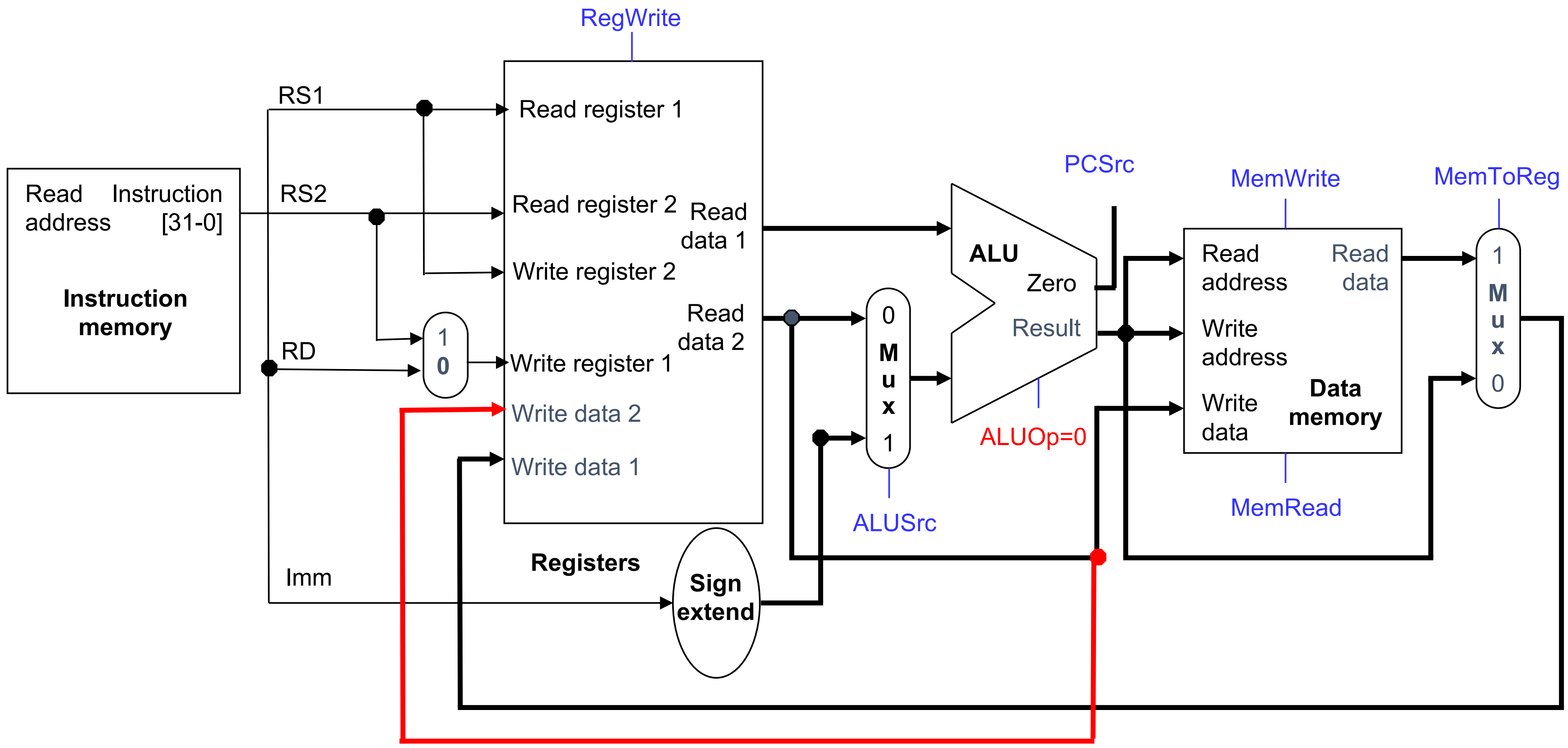
- Original processor has a clock period of 965
- New processor has a clock period of 975
- Total instruction is reduced 0.12*(0.25+0.11) for load and store=0.0432
- CPU time for original processor=N*965
- CPU time for new processor=N*(1-0.0432)*975=N*932.88
- Thus, the speedup would be 965/932.88=1.04

Problem 3-2

- PC + I-Mem + Register File +D-Mem+ Mux*3 + ALU + Adder*2 + Single gate* 2 + Sign extension + Control unit
- Original processor=5 + 1000 + 200+ 2000 + 10*3 + 100 + 30*2 + 1*2 + 100 + 500=3997
- New processor=5 + 1000 + 400+ 2000 + 10*3 + 100 + 30*2 + 1*2 + 100 + 500=4197
- About 5% increase in area cost

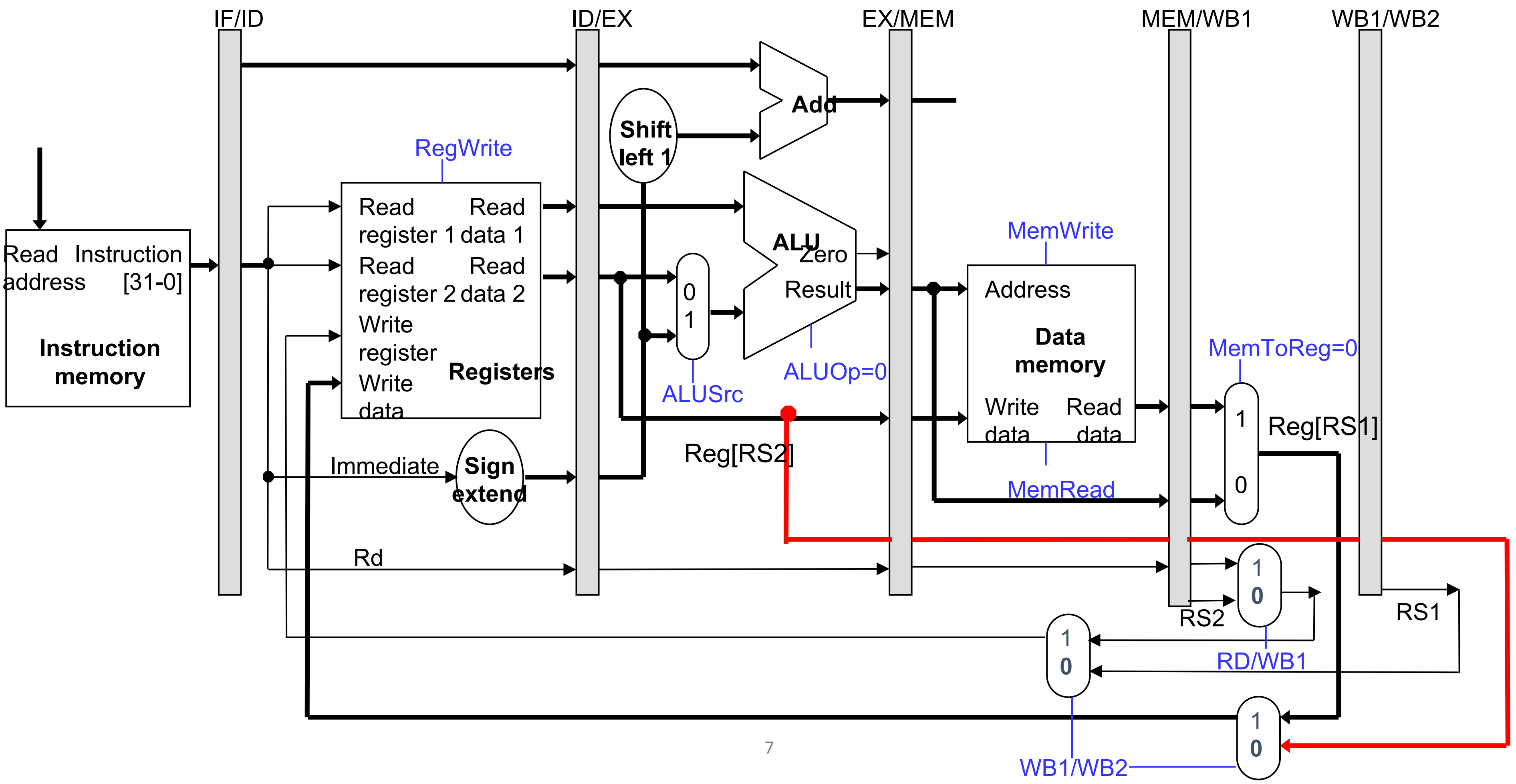
Problem 4-1 Single cycle for swap instruction

- Register file need two write ports; RS1/RS2 routed to write port addresses; ALU result (no operation) and RS2 data are routed to write port data.



Problem 4-1 swap instruction pipeline

- Two cycles are needed for WB: IF ID EX MEM WB1 WB2



Problem 5

- Problem 5-1: 350ps
- Problem 5-2: ID stage split into ID1 and ID2: 350ps/2=175ps
  Cycle time=300ps
- load and store stages use D-MEM: 15+20=35%

Problem 6

Clock cycle

	1	2	3	4	5	6	7	8	9	10	11
`addi x11, x12, 5`	IF	ID	EX	MEM	WB						
`nop`		IF									
`nop`			IF								
`add x13, x11, x12`				IF	ID	EX	MEM	WB			
`addi  x14, x11, 15`					IF	ID	EX	MEM	WB		
`nop`						IF					
`add x15, x13, x12`							IF	ID	EX	MEM	WB

Problem 7

- Problem 7.1
  - CPU time 1 (no forwarding)=1.1N*250ps
  - CPU time 2 (with forwarding)=N*300ps
  - No forwarding speedup ratio=(1*300)/(1.1*250)=1.09

- Problem 7.2
  - Assume x percentage of additional instruction if no forwarding
  - (1+x)*250=300 => x=300/250-1= 20%

Problem 8

	Clock cycle											
	1	2	3	4	5	6	7	8	9	10	11	12
sd x29, 12(x16)	IF	ID	EX	MEM	WB							
ld x29, 8(x16)		IF	ID	EX	MEM	WB						
sub x17, x15, x14			IF	ID	EX	MEM	WB					
bez x17, label				**	**	IF	ID	EX	MEM	WB		
add x15, x11, x14							IF	ID	EX	MEM	WB	
sub x15,x30,x14								IF	ID	EX	MEM	WB

Problem 9

9.2: No cycle has all 5 stages working at the same time.

Clock cycle

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
`ld x10, 0(x13)`	IF	ID	EX	ME	WB											
`ld x11, 8(x13)`		IF	ID	EX	ME	WB										
`add x12, x10, x11`			IF	ID	**	EX	ME!	WB								
`addi x13, x13, -16`				IF		ID	EX	ME!	WB							
`bnez x12, LOOP`						IF	ID	EX	ME!	WB!						
`ld x10, 0(x13)`							IF	ID	EX	ME	WB					
`ld x11, 8(x13)`								IF	ID	EX	ME	WB				
`add x12, x10, x11`									IF	ID	**	EX	ME!	WB		
`addi x13, x13, -16`										IF		ID	EX	ME!	WB	
`bnez x12, LOOP`												IF	ID	EX	ME!	WB!

Problem 10.1

- (1) For processors without forwarding:
```
add x11, x12, x13
nop
nop
add x14, x11, x15
add x5, x6, x7
```
- (2) For processors with forwarding from EX/MEM to following instruction at EX:
```
add x11, x12, x13
add x14, x11, x15
add x5, x6, x7
```
- (3) For processors with forwarding from MEM/WB to following instruction at EX :
```
add x11, x12, x13
nop
add x14, x11, x15
add x5, x6, x7
```
- (4) For processors with full forwarding (both EX/MEM and MEM/WB):
```
add x11, x12, x13
add x14, x11, x15
add x5, x6, x7
```

Problem 10.2

- (1) For processors without forwarding:
```
ld x11, 0(x12)
nop
nop
add x15, x11, x13
add x5, x6, x7
```
- (2) For processors with forwarding from EX/MEM only:
```
ld x11, 0(x12)
nop
nop
add x15, x11, x13
add x5, x6, x7
```
- (3) For processors with forwarding from MEM/WB only:
```
ld x11, 0(x12)
nop
add x15, x11, x13
add x5, x6, x7
```
- (4) For processors with full forwarding (both EX/MEM and MEM/WB):
```
ld x11, 0(x12)
nop
add x15, x11, x13
add x5, x6, x7
```

Problem 10.3

- (1) For processors without forwarding:
```
add x11, x12, x13
add x5, x6, x7
nop
add x14, x11, x12
```
- (2) For processors with forwarding from EX/MEM only:
```
add x11, x12, x13
add x5, x6, x7
nop
add x14, x11, x12
```
- (3) For processors with forwarding from MEM/WB only:
```
add x11, x12, x13
add x5, x6, x7
add x14, x11, x12
```
- (4) For processors with full forwarding (both EX/MEM and MEM/WB):
```
add x11, x12, x13
add x5, x6, x7
add x14, x11, x12
```

Problem 10.4

- (1) For processors without forwarding:
  ```
  ld x11, 0(x12)
  add x5, x6, x7
  nop
  add x14, x11, x13
  ```
- (2) For processors with forwarding from EX/MEM only:
  ```
  ld x11, 0(x12)
  add x5, x6, x7
  nop
  add x14, x11, x13
  ```
- (3) For processors with forwarding from MEM/WB only:
  ```
  ld x11, 0(x12)
  add x5, x6, x7
  add x14, x11, x13
  ```
- (4) For processors with full forwarding (both EX/MEM and MEM/WB):
  ```
  ld x11, 0(x12)
  add x5, x6, x7
  add x14, x11, x13
  ```

Problem 10.5

- (1) For processors without forwarding:
```
add x11, x12, x13
nop
nop
add x5, x11, x15
add x16, x11, x12
```
- (2) For processors with forwarding from EX/MEM only:
```
add x11, x12, x13
add x5, x11, x15
nop
add x16, x11, x12
```
- (3) For processors with forwarding from MEM/WB only:
```
add x11, x12, x13
nop
add x5, x11, x15
add x16, x11, x12
```
- (4) For processors with full forwarding (both EX/MEM and MEM/WB):
```
add x11, x12, x13
add x5, x11, x15
add x16, x11, x12
```

Problem 11

- We label instructions as follows:
  I1: add x15, x12, x11
  I2: ld x13, 4(x15)
  I3: ld x12, 0(x2)
  I4: or x13, x15, x13
  I5: sd x13, 0(x15)
- Forward from I2 at WB to I4 at EX (x13 at RS2)
  The following should be true and then ForwardB = 1
  I2 MEM/WB.RegWrite==1 and
  I2 MEM/WB.rd == I4 ID/EX.rs2 == x13 and
  (I3 EX/MEM.rd == x12 ≠ ID/EX.rs2 == x13 or not(I3 EX/MEM.RegWrite==1))
- Forward from I4 at MEM to I5 at EX (x13 at RS2)
  The following should be true and then ForwardB = 2
  I4 EX/MEM.RegWrite==1 and
  I4 EX/MEM.rd == I5 ID/EX.rs2 == x13