DEPARTMENT OF
NETWORKED SYSTEMS
AND SERVICES

COMPUTER ARCHITECTURES

Practical Tasks in:

Pipeline Scheduling

Gábor Lencse

BUTE Department of Networked Systems and Services
lencse@hit.bme.hu

Budapest,
2023. 05. 18.

MŰEGYETEM 1782

TASK 1

- Consider the following sequence of instructions.

  i1: R0 ← MEM [R1+8]

  i2: R2 ← R0 * 3

  i3: R3 ← MEM [R1+12]

  i4: R4 ← R3 * 5

  i5: R0 ← R2 + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

  a) Identify all the data dependencies among the instructions.

Continued...

TASK 1

b) Schedule the instructions in the pipeline (determine which instruction is in which stage in every time step). If a pipeline stall needs to be inserted, provide the reason by the following notation:
   - D*: the reason of the stall is **D**ata hazard
   - S*: the reason of the stall is **S**tructural hazard
   - C*: the reason of the stall is **C**ontrol hazard

c) For each instruction, decide whether forwarding is necessary in the EX stage, if yes, tell which pipeline register the operand is read from.

d) Reorder the instructions to improve the execution time (while keeping the semantics of the program).

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: **R0** ← MEM [R1+8]

  i2: R2 ← R0 * 3

  i3: R3 ← MEM [R1+12]

  i4: R4 ← R3 * 5

  i5: R0 ← R2 + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

  a) Identify all the data dependencies among the instructions.

  RAW:

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: **R0** ← MEM [R1+8]

  i2: R2 ← **R0** * 3

  i3: R3 ← MEM [R1+12]

  i4: R4 ← R3 * 5

  i5: R0 ← R2 + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2,

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: R0 ← MEM [R1+8]

  i2: **R2** ← R0 * 3

  i3: R3 ← MEM [R1+12]

  i4: R4 ← R3 * 5

  i5: R0 ← R2 + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2,

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: R0 ← MEM [R1+8]

  i2: **R2** ← R0 * 3

  i3: R3 ← MEM [R1+12]

  i4: R4 ← R3 * 5

  i5: R0 ← **R2** + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2, i2 ↔ i5,

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: R0 ← MEM [R1+8]

  i2: R2 ← R0 * 3

  i3: **R3** ← MEM [R1+12]

  i4: R4 ← R3 * 5

  i5: R0 ← R2 + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2, i2 ↔ i5,

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: R0 ← MEM [R1+8]

  i2: R2 ← R0 * 3

  i3: **R3** ← MEM [R1+12]

  i4: R4 ← **R3** * 5

  i5: R0 ← R2 + R4
- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2, i2 ↔ i5, i3 ↔ i4,

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: R0 ← MEM [R1+8]

  i2: R2 ← R0 * 3

  i3: R3 ← MEM [R1+12]

  i4: **R4** ← R3 * 5

  i5: R0 ← R2 + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2, i2 ↔ i5, i3 ↔ i4,

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: R0 ← MEM [R1+8]

  i2: R2 ← R0 * 3

  i3: R3 ← MEM [R1+12]

  i4: **R4** ← R3 * 5

  i5: R0 ← R2 + **R4**

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2, i2 ↔ i5, i3 ↔ i4, i4 ↔ i5

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: R0 ← MEM [R1+8]

  i2: R2 ← R0 * 3

  i3: R3 ← MEM [R1+12]

  i4: R4 ← R3 * 5

  i5: **R0** ← R2 + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2, i2 ↔ i5, i3 ↔ i4, i4 ↔ i5

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: **R0** ← MEM [R1+8]

  i2: R2 ← R0 * 3

  i3: R3 ← MEM [R1+12]

  i4: R4 ← R3 * 5

  i5: R0 ← R2 + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2, i2 ↔ i5, i3 ↔ i4, i4 ↔ i5

WAW:

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: **R0** ← MEM [R1+8]

  i2: R2 ← R0 * 3

  i3: R3 ← MEM [R1+12]

  i4: R4 ← R3 * 5

  i5: **R0** ← R2 + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2, i2 ↔ i5, i3 ↔ i4, i4 ↔ i5

WAW: i1 ↔ i5

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: R0 ← MEM [R1+8]

  i2: **R2** ← R0 * 3

  i3: R3 ← MEM [R1+12]

  i4: R4 ← R3 * 5

  i5: R0 ← R2 + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2, i2 ↔ i5, i3 ↔ i4, i4 ↔ i5

WAW: i1 ↔ i5

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: R0 ← MEM [R1+8]

  i2: R2 ← R0 * 3

  i3: **R3** ← MEM [R1+12]

  i4: R4 ← R3 * 5

  i5: R0 ← R2 + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2, i2 ↔ i5, i3 ↔ i4, i4 ↔ i5

WAW: i1 ↔ i5

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: R0 ← MEM [R1+8]

  i2: R2 ← R0 * 3

  i3: R3 ← MEM [R1+12]

  i4: **R4** ← R3 * 5

  i5: R0 ← R2 + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2, i2 ↔ i5, i3 ↔ i4, i4 ↔ i5

WAW: i1 ↔ i5

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: R0 ← MEM [R1+8]

  i2: R2 ← R0 * 3

  i3: R3 ← MEM [R1+12]

  i4: R4 ← R3 * 5

  i5: **R0** ← R2 + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2, i2 ↔ i5, i3 ↔ i4, i4 ↔ i5

WAW: i1 ↔ i5

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: R0 ← MEM [**R1**+8]

  i2: R2 ← R0 * 3

  i3: R3 ← MEM [R1+12]

  i4: R4 ← R3 * 5

  i5: R0 ← R2 + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2, i2 ↔ i5, i3 ↔ i4, i4 ↔ i5

WAW: i1 ↔ i5

WAR:

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: R0 ← MEM [R1+8]

  i2: R2 ← **R0** * 3

  i3: R3 ← MEM [R1+12]

  i4: R4 ← R3 * 5

  i5: R0 ← R2 + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2, i2 ↔ i5, i3 ↔ i4, i4 ↔ i5

WAW: i1 ↔ i5

WAR:

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: R0 ← MEM [R1+8]

  i2: R2 ← **R0** * 3

  i3: R3 ← MEM [R1+12]

  i4: R4 ← R3 * 5

  i5: **R0** ← R2 + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2, i2 ↔ i5, i3 ↔ i4, i4 ↔ i5

WAW: i1 ↔ i5

WAR: i2 ↔ i5

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: R0 ← MEM [R1+8]

  i2: R2 ← R0 * 3

  i3: R3 ← MEM [**R1**+12]

  i4: R4 ← R3 * 5

  i5: R0 ← R2 + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2, i2 ↔ i5, i3 ↔ i4, i4 ↔ i5

WAW: i1 ↔ i5

WAR: i2 ↔ i5

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: R0 ← MEM [R1+8]

  i2: R2 ← R0 * 3

  i3: R3 ← MEM [R1+12]

  i4: R4 ← **R3** * 5

  i5: R0 ← R2 + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2, i2 ↔ i5, i3 ↔ i4, i4 ↔ i5

WAW: i1 ↔ i5

WAR: i2 ↔ i5

TASK 1 – SOLUTION

- Consider the following sequence of instructions.

  i1: R0 ← MEM [R1+8]

  i2: R2 ← R0 * 3

  i3: R3 ← MEM [R1+12]

  i4: R4 ← R3 * 5

  i5: R0 ← R2 + R4

- These instructions are executed on a CPU which uses the simple 5-stage pipeline studied in the lecture.

a) Identify all the data dependencies among the instructions.

RAW: i1 ↔ i2, i2 ↔ i5, i3 ↔ i4, i4 ↔ i5

WAW: i1 ↔ i5

WAR: i2 ↔ i5 (Ready)

TASK 1 – SOLUTION

b) Schedule the instructions in the pipeline (determine which instruction is in which stage in every time step). If a pipeline stall needs to be inserted, provide the reason by the following notation:
   - D*: the reason of the stall is **D**ata hazard
   - S*: the reason of the stall is **S**tructural hazard
   - C*: the reason of the stall is **C**ontrol hazard

c) For each instruction, decide whether forwarding is necessary in the EX stage, if yes, tell which pipeline register the operand is read from.

We shall solve the above two sub-tasks together.

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]											
R2 ← R0 * 3											
R3 ← MEM [R1+12]											
R4 ← R3 * 5											
R0 ← R2 + R4											

Is forwarding needed in the EX stage?

- i1:
- i2:
- i3:
- i4:
- i5:

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]	IF	ID	EX	MEM	WB						
R2 ← R0 * 3											
R3 ← MEM [R1+12]											
R4 ← R3 * 5											
R0 ← R2 + R4											

Is forwarding needed in the EX stage?

- i1:
- i2:
- i3:
- i4:
- i5:

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]	IF	ID	EX	MEM	WB						
R2 ← R0 * 3											
R3 ← MEM [R1+12]											
R4 ← R3 * 5											
R0 ← R2 + R4											

Is forwarding needed in the EX stage?

- i1: no (trivial)
- i2:
- i3:
- i4:
- i5:

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]	IF	ID	EX	MEM	WB						
R2 ← R0 * 3		IF	ID								
R3 ← MEM [R1+12]											
R4 ← R3 * 5											
R0 ← R2 + R4											

Is forwarding needed in the EX stage?

- i1: no (trivial)
- i2:
- i3:
- i4:
- i5:

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]	IF	ID	EX	MEM	WB						
R2 ← R0 * 3		IF	ID	D*							
R3 ← MEM [R1+12]											
R4 ← R3 * 5											
R0 ← R2 + R4											

Is forwarding needed in the EX stage?

- i1: no (trivial)
- i2:
- i3:
- i4:
- i5:

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]	IF	ID	EX	MEM	WB						
R2 ← R0 * 3		IF	ID	D*	EX	MEM	WB				
R3 ← MEM [R1+12]											
R4 ← R3 * 5											
R0 ← R2 + R4											

Is forwarding needed in the EX stage?

- i1: no (trivial)
- i2:
- i3:
- i4:
- i5:

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]	IF	ID	EX	MEM	WB						
R2 ← R0 * 3		IF	ID	D*	EX	MEM	WB				
R3 ← MEM [R1+12]											
R4 ← R3 * 5											
R0 ← R2 + R4											

Is forwarding needed in the EX stage?

- i1: no (trivial)
- i2: yes: R0 is taken from pipeline register MEM/WB
- i3:
- i4:
- i5:

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]	IF	ID	EX	MEM	WB						
R2 ← R0 * 3		IF	ID	D*	EX	MEM	WB				
R3 ← MEM [R1+12]			IF								
R4 ← R3 * 5											
R0 ← R2 + R4											

Is forwarding needed in the EX stage?

- i1: no (trivial)
- i2: yes: R0 is taken from pipeline register MEM/WB
- i3:
- i4:
- i5:

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]	IF	ID	EX	MEM	WB						
R2 ← R0 * 3		IF	ID	D*	EX	MEM	WB				
R3 ← MEM [R1+12]			IF	S*							
R4 ← R3 * 5											
R0 ← R2 + R4											

Is forwarding needed in the EX stage?

- i1: no (trivial)
- i2: yes: R0 is taken from pipeline register MEM/WB
- i3:
- i4:
- i5:

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]	IF	ID	EX	MEM	WB						
R2 ← R0 * 3		IF	ID	D*	EX	MEM	WB				
R3 ← MEM [R1+12]			IF	S*	ID	EX	MEM	WB			
R4 ← R3 * 5											
R0 ← R2 + R4											

Is forwarding needed in the EX stage?

- i1: no (trivial)
- i2: yes: R0 is taken from pipeline register MEM/WB
- i3:
- i4:
- i5:

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]	IF	ID	EX	MEM	WB						
R2 ← R0 * 3		IF	ID	D*	EX	MEM	WB				
R3 ← MEM [R1+12]			IF	S*	ID	EX	MEM	WB			
R4 ← R3 * 5											
R0 ← R2 + R4											

Is forwarding needed in the EX stage?

- i1: no (trivial)
- i2: yes: R0 is taken from pipeline register MEM/WB
- i3: no
- i4:
- i5:

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]	IF	ID	EX	MEM	WB						
R2 ← R0 * 3		IF	ID	D*	EX	MEM	WB				
R3 ← MEM [R1+12]			IF	S*	ID	EX	MEM	WB			
R4 ← R3 * 5					IF	ID					
R0 ← R2 + R4											

Is forwarding needed in the EX stage?

- i1: no (trivial)
- i2: yes: R0 is taken from pipeline register MEM/WB
- i3: no
- i4:
- i5:

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]	IF	ID	EX	MEM	WB						
R2 ← R0 * 3		IF	ID	D*	EX	MEM	WB				
R3 ← MEM [R1+12]			IF	S*	ID	EX	MEM	WB			
R4 ← R3 * 5					IF	ID	D*				
R0 ← R2 + R4											

Is forwarding needed in the EX stage?

- i1: no (trivial)
- i2: yes: R0 is taken from pipeline register MEM/WB
- i3: no
- i4:
- i5:

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]	IF	ID	EX	MEM	WB						
R2 ← R0 * 3		IF	ID	D*	EX	MEM	WB				
R3 ← MEM [R1+12]			IF	S*	ID	EX	MEM	WB			
R4 ← R3 * 5					IF	ID	D*	EX	MEM	WB	
R0 ← R2 + R4											

Is forwarding needed in the EX stage?

- i1: no (trivial)
- i2: yes: R0 is taken from pipeline register MEM/WB
- i3: no
- i4:
- i5:

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]	IF	ID	EX	MEM	WB						
R2 ← R0 * 3		IF	ID	D*	EX	MEM	WB				
R3 ← MEM [R1+12]			IF	S*	ID	EX	MEM	WB			
R4 ← R3 * 5					IF	ID	D*	EX	MEM	WB	
R0 ← R2 + R4											

Is forwarding needed in the EX stage?

- i1: no (trivial)
- i2: yes: R0 is taken from pipeline register MEM/WB
- i3: no
- i4: yes: R3 is taken from pipeline register MEM/WB
- i5:

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]	IF	ID	EX	MEM	WB						
R2 ← R0 * 3		IF	ID	D*	EX	MEM	WB				
R3 ← MEM [R1+12]			IF	S*	ID	EX	MEM	WB			
R4 ← R3 * 5					IF	ID	D*	EX	MEM	WB	
R0 ← R2 + R4						IF					

Is forwarding needed in the EX stage?

- i1: no (trivial)
- i2: yes: R0 is taken from pipeline register MEM/WB
- i3: no
- i4: yes: R3 is taken from pipeline register MEM/WB
- i5:

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]	IF	ID	EX	MEM	WB						
R2 ← R0 * 3		IF	ID	D*	EX	MEM	WB				
R3 ← MEM [R1+12]			IF	S*	ID	EX	MEM	WB			
R4 ← R3 * 5					IF	ID	D*	EX	MEM	WB	
R0 ← R2 + R4						IF	S*				

Is forwarding needed in the EX stage?

- i1: no (trivial)
- i2: yes: R0 is taken from pipeline register MEM/WB
- i3: no
- i4: yes: R3 is taken from pipeline register MEM/WB
- i5:

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]	IF	ID	EX	MEM	WB						
R2 ← R0 * 3		IF	ID	D*	EX	MEM	WB				
R3 ← MEM [R1+12]			IF	S*	ID	EX	MEM	WB			
R4 ← R3 * 5					IF	ID	D*	EX	MEM	WB	
R0 ← R2 + R4						IF	S*	ID	EX	MEM	WB

Is forwarding needed in the EX stage?

- i1: no (trivial)
- i2: yes: R0 is taken from pipeline register MEM/WB
- i3: no
- i4: yes: R3 is taken from pipeline register MEM/WB
- i5:

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]	IF	ID	EX	MEM	WB						
R2 ← R0 * 3		IF	ID	D*	EX	MEM	WB				
R3 ← MEM [R1+12]			IF	S*	ID	EX	MEM	WB			
R4 ← R3 * 5					IF	ID	D*	EX	MEM	WB	
R0 ← R2 + R4						IF	S*	ID	EX	MEM	WB

Is forwarding needed in the EX stage?

- i1: no (trivial)
- i2: yes: R0 is taken from pipeline register MEM/WB
- i3: no
- i4: yes: R3 is taken from pipeline register MEM/WB
- i5: yes: R4 is taken from pipeline register EX/MEM

TASK 1 – SOLUTION

d) Reorder the instructions to improve the execution time (while keeping the semantics of the program)

- What shall we do?
  - The pipeline was stalled twice, due to RAW (i1 ↔ i2, i3 ↔ i4)
  - Let us move these instructions away from each other!
  - We need to exchange i2 and i3.

Old code:	Optimized code:
i1: R0 ← MEM [R1+8]	R0 ← MEM [R1+8]
i2: R2 ← R0 * 3	R3 ← MEM [R1+12]
i3: R3 ← MEM [R1+12]	R2 ← R0 * 3
i4: R4 ← R3 * 5	R4 ← R3 * 5
i5: R0 ← R2 + R4	R0 ← R2 + R4

- Although not required, we shall prepare the scheduling, too.

TASK 1 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
R0 ← MEM [R1+8]	IF	ID	EX	MEM	WB						
R3 ← MEM [R1+12]		IF	ID	EX	MEM	WB					
R2 ← R0 * 3			IF	ID	EX	MEM	WB				
R4 ← R3 * 5				IF	ID	EX	MEM	WB			
R0 ← R2 + R4					IF	ID	EX	MEM	WB		

Is forwarding needed in the EX stage?

- i1: no (trivial)
- i2: no
- i3: yes: R0 is taken from pipeline register MEM/WB
- i4: yes: R3 is taken from pipeline register MEM/WB
- i5: yes: R2 is taken from pipeline register MEM/WB
  R4 is taken from pipeline register EX/MEM

TASK 2

- Consider the following sequence of instructions.

  i1: R5 ← MEM [R3+24]

  i2: R6 ← MEM [R4+16]

  i3: R7 ← R6 + R5

  i4: R8 ← R6 - R5

  i5: R5 ← R7 * R8

  i6: R4 ← R4 + 4

- These instructions are executed on a CPU which uses a 6-stage pipeline. The execution of the instructions consists of 5 phases: fetch (IF), decode (ID), execute arithmetic-logic operations (EX), perform memory operations (MEM), and writing back the result into the register file (WB). The latency of the ID, EX, MEM, WB phases is 1 clock cycle, while the latency of the IF phase is 2 clock cycles, the iteration interval is still 1. This means that the instruction fetch phase consists of two stages: IF0 and IF1.

TASK 2

a) Identify all the data dependencies among the instructions
b) Schedule the instructions in the pipeline (determine which instruction is in which stage in every time step). If a pipeline stall needs to be inserted, provide the reason by the following notation:
   - D*: the reason of the stall is data hazard
   - S*: the reason of the stall is structural hazard
   - C*: the reason of the stall is control hazard
c) For each instruction, decide whether forwarding is necessary in the EX stage, if yes, tell which pipeline register the operand is read from
d) Reorder the instructions to improve the execution time (while keeping the semantics of the program)

TASK 2 – SOLUTION

i1: **R5** ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: R8 ← R6 - R5

i5: R5 ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW:

TASK 2 – SOLUTION

i1: **R5** ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← R6 + **R5**

i4: R8 ← R6 - R5

i5: R5 ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3,

TASK 2 – SOLUTION

i1: **R5** ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: R8 ← R6 - **R5**

i5: R5 ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4,

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: **R6** ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: R8 ← R6 - R5

i5: R5 ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4,

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: **R6** ← MEM [R4+16]

i3: R7 ← **R6** + R5

i4: R8 ← R6 - R5

i5: R5 ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3,

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: **R6** ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: R8 ← **R6** - R5

i5: R5 ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4,

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: **R7** ← R6 + R5

i4: R8 ← R6 - R5

i5: R5 ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4,

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: **R7** ← R6 + R5

i4: R8 ← R6 - R5

i5: R5 ← **R7** * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5,

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: **R8** ← R6 - R5

i5: R5 ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5,

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: **R8** ← R6 - R5

i5: R5 ← R7 * **R8**

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5, i4 ↔ i5

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: R8 ← R6 - R5

i5: **R5** ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5, i4 ↔ i5

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: R8 ← R6 - R5

i5: R5 ← R7 * R8

i6: **R4** ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5, i4 ↔ i5

TASK 2 – SOLUTION

i1: **R5** ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: R8 ← R6 - R5

i5: R5 ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5, i4 ↔ i5

WAW:

TASK 2 – SOLUTION

i1: **R5** ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: R8 ← R6 - R5

i5: **R5** ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5, i4 ↔ i5

WAW: i1 ↔ i5

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: **R6** ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: R8 ← R6 - R5

i5: R5 ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5, i4 ↔ i5

WAW: i1 ↔ i5

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: **R7** ← R6 + R5

i4: R8 ← R6 - R5

i5: R5 ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5, i4 ↔ i5

WAW: i1 ↔ i5

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: **R8** ← R6 - R5

i5: R5 ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5, i4 ↔ i5

WAW: i1 ↔ i5

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: R8 ← R6 - R5

i5: **R5** ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5, i4 ↔ i5

WAW: i1 ↔ i5

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: R8 ← R6 - R5

i5: R5 ← R7 * R8

i6: **R4** ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5, i4 ↔ i5

WAW: i1 ↔ i5

TASK 2 – SOLUTION

i1: R5 ← MEM [**R3**+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: R8 ← R6 - R5

i5: R5 ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5, i4 ↔ i5

WAW: i1 ↔ i5

WAR:

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: R6 ← MEM [**R4**+16]

i3: R7 ← R6 + R5

i4: R8 ← R6 - R5

i5: R5 ← R7 * R8

i6: **R4** ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5, i4 ↔ i5

WAW: i1 ↔ i5

WAR: i2 ↔ i6,

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← **R6** + R5

i4: R8 ← R6 - R5

i5: R5 ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5, i4 ↔ i5

WAW: i1 ↔ i5

WAR: i2 ↔ i6,

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← R6 + **R5**

i4: R8 ← R6 - R5

i5: **R5** ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5, i4 ↔ i5

WAW: i1 ↔ i5

WAR: i2 ↔ i6, i3 ↔ i5,

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: R8 ← **R6** - R5

i5: R5 ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5, i4 ↔ i5

WAW: i1 ↔ i5

WAR: i2 ↔ i6, i3 ↔ i5,

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: R8 ← R6 - **R5**

i5: **R5** ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5, i4 ↔ i5

WAW: i1 ↔ i5

WAR: i2 ↔ i6, i3 ↔ i5, i4 ↔ i5

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: R8 ← R6 - R5

i5: R5 ← **R7** * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5, i4 ↔ i5

WAW: i1 ↔ i5

WAR: i2 ↔ i6, i3 ↔ i5, i4 ↔ i5

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: R8 ← R6 - R5

i5: R5 ← R7 * **R8**

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5, i4 ↔ i5

WAW: i1 ↔ i5

WAR: i2 ↔ i6, i3 ↔ i5, i4 ↔ i5

TASK 2 – SOLUTION

i1: R5 ← MEM [R3+24]

i2: R6 ← MEM [R4+16]

i3: R7 ← R6 + R5

i4: R8 ← R6 - R5

i5: R5 ← R7 * R8

i6: R4 ← R4 + 4

a) Identification of the dependencies

RAW: i1 ↔ i3, i1 ↔ i4, i2 ↔ i3, i2 ↔ i4, i3 ↔ i5, i4 ↔ i5

WAW: i1 ↔ i5

WAR: i2 ↔ i6, i3 ↔ i5, i4 ↔ i5

(Ready)

TASK 2 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
R5 ← MEM[R3+24]												
R6 ← MEM[R4+16]												
R7 ← R6 + R5												
R8 ← R6 - R5												
R5 ← R7 * R8												
R4 ← R4 + 4												

Is forwarding needed in the EX stage?

- i2:
- i3:
- i4:
- i5:

- i6:

TASK 2 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
R5 ← MEM[R3+24]	IF0	IF1	ID	EX	MEM	WB						
R6 ← MEM[R4+16]												
R7 ← R6 + R5												
R8 ← R6 - R5												
R5 ← R7 * R8												
R4 ← R4 + 4												

Is forwarding needed in the EX stage?

- i2:
- i3:
- i4:
- i5:

- i6:

TASK 2 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
R5 ← MEM[R3+24]	IF0	IF1	ID	EX	MEM	WB						
R6 ← MEM[R4+16]		IF0	IF1	ID	EX	MEM	WB					
R7 ← R6 + R5												
R8 ← R6 - R5												
R5 ← R7 * R8												
R4 ← R4 + 4												

Is forwarding needed in the EX stage?

- i2:
- i3:
- i4:
- i5:

- i6:

TASK 2 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
R5 ← MEM[R3+24]	IF0	IF1	ID	EX	MEM	WB						
R6 ← MEM[R4+16]		IF0	IF1	ID	EX	MEM	WB					
R7 ← R6 + R5												
R8 ← R6 - R5												
R5 ← R7 * R8												
R4 ← R4 + 4												

Is forwarding needed in the EX stage?

- i2: no
- i3:
- i4:
- i5:

- i6:

TASK 2 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
R5 ← MEM[R3+24]	IF0	IF1	ID	EX	MEM	WB						
R6 ← MEM[R4+16]		IF0	IF1	ID	EX	MEM	WB					
R7 ← R6 + R5			IF0	IF1	ID							
R8 ← R6 - R5												
R5 ← R7 * R8												
R4 ← R4 + 4												

Is forwarding needed in the EX stage?

- i2: no
- i3:
- i4:
- i5:

- i6:

TASK 2 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
R5 ← MEM[R3+24]	IF0	IF1	ID	EX	MEM	WB						
R6 ← MEM[R4+16]		IF0	IF1	ID	EX	MEM	WB					
R7 ← R6 + R5			IF0	IF1	ID	D*						
R8 ← R6 - R5												
R5 ← R7 * R8												
R4 ← R4 + 4												

Is forwarding needed in the EX stage?

- i2: no
- i3:
- i4:
- i5:

- i6:

TASK 2 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
R5 ← MEM[R3+24]	IF0	IF1	ID	EX	MEM	WB						
R6 ← MEM[R4+16]		IF0	IF1	ID	EX	MEM	WB					
R7 ← R6 + R5			IF0	IF1	ID	D*	EX	MEM	WB			
R8 ← R6 - R5												
R5 ← R7 * R8												
R4 ← R4 + 4												

Is forwarding needed in the EX stage?

- i2: no
- i3:
- i4:
- i5:

- i6:

TASK 2 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
R5 ← MEM[R3+24]	IF0	IF1	ID	EX	MEM	WB						
R6 ← MEM[R4+16]		IF0	IF1	ID	EX	MEM	WB					
R7 ← R6 + R5			IF0	IF1	ID	D*	EX	MEM	WB			
R8 ← R6 - R5												
R5 ← R7 * R8												
R4 ← R4 + 4												

Is forwarding needed in the EX stage?

- i2: no
- i3: yes: R6 is taken from pipeline register MEM/WB
- i4:
- i5:

- i6:

TASK 2 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
R5 ← MEM[R3+24]	IF0	IF1	ID	EX	MEM	WB						
R6 ← MEM[R4+16]		IF0	IF1	ID	EX	MEM	WB					
R7 ← R6 + R5			IF0	IF1	ID	D*	EX	MEM	WB			
R8 ← R6 - R5				IF0	IF1							
R5 ← R7 * R8												
R4 ← R4 + 4												

Is forwarding needed in the EX stage?

- i2: no
- i3: yes: R6 is taken from pipeline register MEM/WB
- i4:
- i5:

- i6:

TASK 2 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
R5 ← MEM[R3+24]	IF0	IF1	ID	EX	MEM	WB						
R6 ← MEM[R4+16]		IF0	IF1	ID	EX	MEM	WB					
R7 ← R6 + R5			IF0	IF1	ID	D*	EX	MEM	WB			
R8 ← R6 - R5				IF0	IF1	S*						
R5 ← R7 * R8												
R4 ← R4 + 4												

Is forwarding needed in the EX stage?

- i2: no
- i3: yes: R6 is taken from pipeline register MEM/WB
- i4:
- i5:

- i6:

TASK 2 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
R5 ← MEM[R3+24]	IF0	IF1	ID	EX	MEM	WB						
R6 ← MEM[R4+16]		IF0	IF1	ID	EX	MEM	WB					
R7 ← R6 + R5			IF0	IF1	ID	D*	EX	MEM	WB			
R8 ← R6 - R5				IF0	IF1	S*	ID	EX	MEM	WB		
R5 ← R7 * R8												
R4 ← R4 + 4												

Is forwarding needed in the EX stage?

- i2: no
- i3: yes: R6 is taken from pipeline register MEM/WB
- i4:
- i5:

- i6:

TASK 2 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
R5 ← MEM[R3+24]	IF0	IF1	ID	EX	MEM	WB						
R6 ← MEM[R4+16]		IF0	IF1	ID	EX	MEM	WB					
R7 ← R6 + R5			IF0	IF1	ID	D*	EX	MEM	WB			
R8 ← R6 - R5				IF0	IF1	S*	ID	EX	MEM	WB		
R5 ← R7 * R8												
R4 ← R4 + 4												

Is forwarding needed in the EX stage?

- i2: no
- i3: yes: R6 is taken from pipeline register MEM/WB
- i4: no
- i5:

- i6:

TASK 2 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
R5 ← MEM[R3+24]	IF0	IF1	ID	EX	MEM	WB						
R6 ← MEM[R4+16]		IF0	IF1	ID	EX	MEM	WB					
R7 ← R6 + R5			IF0	IF1	ID	D*	EX	MEM	WB			
R8 ← R6 - R5				IF0	IF1	S*	ID	EX	MEM	WB		
R5 ← R7 * R8					IF0							
R4 ← R4 + 4												

Is forwarding needed in the EX stage?

- i2: no
- i3: yes: R6 is taken from pipeline register MEM/WB
- i4: no
- i5:

- i6:

TASK 2 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
R5 ← MEM[R3+24]	IF0	IF1	ID	EX	MEM	WB						
R6 ← MEM[R4+16]		IF0	IF1	ID	EX	MEM	WB					
R7 ← R6 + R5			IF0	IF1	ID	D*	EX	MEM	WB			
R8 ← R6 - R5				IF0	IF1	S*	ID	EX	MEM	WB		
R5 ← R7 * R8					IF0	S*						
R4 ← R4 + 4												

Is forwarding needed in the EX stage?

- i2: no
- i3: yes: R6 is taken from pipeline register MEM/WB
- i4: no
- i5:

- i6:

TASK 2 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
R5 ← MEM[R3+24]	IF0	IF1	ID	EX	MEM	WB						
R6 ← MEM[R4+16]		IF0	IF1	ID	EX	MEM	WB					
R7 ← R6 + R5			IF0	IF1	ID	D*	EX	MEM	WB			
R8 ← R6 - R5				IF0	IF1	S*	ID	EX	MEM	WB		
R5 ← R7 * R8					IF0	S*	IF1	ID				
R4 ← R4 + 4												

Is forwarding needed in the EX stage?

- i2: no
- i3: yes: R6 is taken from pipeline register MEM/WB
- i4: no
- i5:

- i6:

TASK 2 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
R5 ← MEM[R3+24]	IF0	IF1	ID	EX	MEM	WB						
R6 ← MEM[R4+16]		IF0	IF1	ID	EX	MEM	WB					
R7 ← R6 + R5			IF0	IF1	ID	D*	EX	MEM	WB			
R8 ← R6 - R5				IF0	IF1	S*	ID	EX	MEM	WB		
R5 ← R7 * R8					IF0	S*	IF1	ID	EX	MEM	WB	
R4 ← R4 + 4												

Is forwarding needed in the EX stage?

- i2: no
- i3: yes: R6 is taken from pipeline register MEM/WB
- i4: no
- i5:

- i6:

TASK 2 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
R5 ← MEM[R3+24]	IF0	IF1	ID	EX	MEM	WB						
R6 ← MEM[R4+16]		IF0	IF1	ID	EX	MEM	WB					
R7 ← R6 + R5			IF0	IF1	ID	D*	EX	MEM	WB			
R8 ← R6 - R5				IF0	IF1	S*	ID	EX	MEM	WB		
R5 ← R7 * R8					IF0	S*	IF1	ID	EX	MEM	WB	
R4 ← R4 + 4												

Is forwarding needed in the EX stage?

- i2: no
- i3: yes: R6 is taken from pipeline register MEM/WB
- i4: no
- i5: yes: R7 is taken from pipeline register MEM/WB
  R8 is taken from pipeline register EX/MEM
- i6:

TASK 2 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
R5 ← MEM[R3+24]	IF0	IF1	ID	EX	MEM	WB						
R6 ← MEM[R4+16]		IF0	IF1	ID	EX	MEM	WB					
R7 ← R6 + R5			IF0	IF1	ID	D*	EX	MEM	WB			
R8 ← R6 - R5				IF0	IF1	S*	ID	EX	MEM	WB		
R5 ← R7 * R8					IF0	S*	IF1	ID	EX	MEM	WB	
R4 ← R4 + 4							IF0	IF1	ID	EX	MEM	WB

Is forwarding needed in the EX stage?

- i2: no
- i3: yes: R6 is taken from pipeline register MEM/WB
- i4: no
- i5: yes: R7 is taken from pipeline register MEM/WB
  R8 is taken from pipeline register EX/MEM
- i6:

TASK 2 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
R5 ← MEM[R3+24]	IF0	IF1	ID	EX	MEM	WB						
R6 ← MEM[R4+16]		IF0	IF1	ID	EX	MEM	WB					
R7 ← R6 + R5			IF0	IF1	ID	D*	EX	MEM	WB			
R8 ← R6 - R5				IF0	IF1	S*	ID	EX	MEM	WB		
R5 ← R7 * R8					IF0	S*	IF1	ID	EX	MEM	WB	
R4 ← R4 + 4							IF0	IF1	ID	EX	MEM	WB

Is forwarding needed in the EX stage?

- i2: no
- i3: yes: R6 is taken from pipeline register MEM/WB
- i4: no
- i5: yes: R7 is taken from pipeline register MEM/WB
  R8 is taken from pipeline register EX/MEM
- i6: no

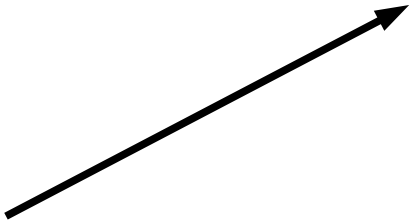
TASK 2 – SOLUTION

d) Reorder the instructions to improve the execution time (while keeping the semantics of the program)

- What shall we do?
  - The pipeline was stalled due to RAW (i2 ↔ i3)
  - Let us move these instructions away from each other!
  - We need to insert i6 before i3.

Old code:

i1: R5 ← MEM [R3+24]
i2: R6 ← MEM [R4+16]
i3: R7 ← R6 + R5
i4: R8 ← R6 – R5
i5: R5 ← R7 * R8
i6: R4 ← R4 + 4



Optimized code:

R5 ← MEM [R3+24]
R6 ← MEM [R4+16]
R4 ← R4 + 4
R7 ← R6 + R5
R8 ← R6 – R5
R5 ← R7 * R8

TASK 3

- Consider the following sequence of instructions.

  i1: D0 ← D1 * D2

  i2: D3 ← D0 + D5

  i3: MEM [R0+4] ← D3

  i4: MEM [R0+12] ← D0

- The D0-D5 registers store floating point numbers, the R0 stores integer number. These instructions are executed on a CPU which uses a 5-stage pipeline. The execution of the instructions consists of 5 phases: fetch (IF), decode (ID), execute arithmetic-logic operations, perform memory operations (MEM), and writing back the result into the register file (WB). An arithmetic operation can be an integer operation (EX, latency is 1), a floating point addition operation (latency is 4, iteration interval is 1, A0, A1, A2, A3) or a floating point multiplication operation (latency is 7, iteration interval is 1, M0, M1, M2, M3, M4, M5, M6). These three operations can be performed in parallel, if there are no hazards.

TASK 3

a) Identify all the data dependencies among the instructions

b) Schedule the instructions in the pipeline (determine which instruction is in which stage in every time step). If a pipeline stall needs to be inserted, provide the reason by the following notation:
   - D*: the reason of the stall is data hazard
   - S*: the reason of the stall is structural hazard
   - C*: the reason of the stall is control hazard

c) For each instruction, decide whether forwarding is necessary in the EX stage, if yes, tell which pipeline register the operand is read from

d) Reorder the instructions to improve the execution time (while keeping the semantics of the program)

TASK 3 – SOLUTION

i1: **D0** ← D1 * D2

i2: D3 ← D0 + D5

i3: MEM [R0+4] ← D3

i4: MEM [R0+12] ← D0

a) Identification of the data dependencies among the instructions

RAW:

WAW:

WAR:

TASK 3 – SOLUTION

i1: **D0** ← D1 * D2

i2: D3 ← **D0** + D5

i3: MEM [R0+4] ← D3

i4: MEM [R0+12] ← D0

a) Identification of the data dependencies among the instructions

RAW: i1 ↔ i2,

WAW:

WAR:

TASK 3 – SOLUTION

i1: **D0** ← D1 * D2

i2: D3 ← D0 + D5

i3: MEM [R0+4] ← D3

i4: MEM [R0+12] ← **D0**

a) Identification of the data dependencies among the instructions

RAW: i1 ↔ i2, i1 ↔ i4,

WAW:

WAR:

TASK 3 – SOLUTION

i1: D0 ← D1 * D2

i2: **D3** ← D0 + D5

i3: MEM [R0+4] ← **D3**

i4: MEM [R0+12] ← D0

a) Identification of the data dependencies among the instructions

RAW: i1 ↔ i2, i1 ↔ i4, i2 ↔ i3

WAW:

WAR:

TASK 3 – SOLUTION

i1: D0 ← D1 * D2

i2: D3 ← D0 + D5

i3: MEM [R0+4] ← D3

i4: MEM [R0+12] ← D0

a) Identification of the data dependencies among the instructions

RAW: i1 ↔ i2, i1 ↔ i4, i2 ↔ i3

WAW: -

WAR:

TASK 3 – SOLUTION

i1: D0 ← D1 * D2

i2: D3 ← D0 + D5

i3: MEM [R0+4] ← D3

i4: MEM [R0+12] ← D0

a) Identification of the data dependencies among the instructions

RAW: i1 ↔ i2, i1 ↔ i4, i2 ↔ i3

WAW: -

WAR: - (Ready)

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2																	
D3 ← D0 + D5																	
MEM[R0+4] ← D3																	
MEM[R0+12] ← D0																	

- Is forwarding needed in the EX stage?
  - i2:
  - i3:
  - i4:

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2	IF	ID															
D3 ← D0 + D5																	
MEM[R0+4] ← D3																	
MEM[R0+12] ← D0																	

- Is forwarding needed in the EX stage?
  - i2:
  - i3:
  - i4:

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2	IF	ID	M0	M1	M2	M3	M4	M5	M6								
D3 ← D0 + D5																	
MEM[R0+4] ← D3																	
MEM[R0+12] ← D0																	

- Is forwarding needed in the EX stage?
  - i2:
  - i3:
  - i4:

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB						
D3 ← D0 + D5																	
MEM[R0+4] ← D3																	
MEM[R0+12] ← D0																	

- Is forwarding needed in the EX stage?
  - i2:
  - i3:
  - i4:

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB						
D3 ← D0 + D5		IF	ID														
MEM[R0+4] ← D3																	
MEM[R0+12] ← D0																	

- Is forwarding needed in the EX stage?
  - i2:
  - i3:
  - i4:

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB						
D3 ← D0 + D5		IF	ID	D*	D*	D*	D*	D*	D*								
MEM[R0+4] ← D3																	
MEM[R0+12] ← D0																	

- Is forwarding needed in the EX stage?
  - i2:
  - i3:
  - i4:

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB						
D3 ← D0 + D5		IF	ID	D*	D*	D*	D*	D*	D*	A0	A1	A2	A3				
MEM[R0+4] ← D3																	
MEM[R0+12] ← D0																	

- Is forwarding needed in the EX stage?
  - i2:
  - i3:
  - i4:

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB						
D3 ← D0 + D5		IF	ID	D*	D*	D*	D*	D*	D*	A0	A1	A2	A3	MEM	WB		
MEM[R0+4] ← D3																	
MEM[R0+12] ← D0																	

- Is forwarding needed in the EX stage?
  - i2:
  - i3:
  - i4:

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB						
D3 ← D0 + D5		IF	ID	D*	D*	D*	D*	D*	D*	A0	A1	A2	A3	MEM	WB		
MEM[R0+4] ← D3																	
MEM[R0+12] ← D0																	

- Is forwarding needed in the EX stage?
  - i2: yes: D0 is taken from pipeline register M6/MEM
  - i3:
  - i4:

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB						
D3 ← D0 + D5		IF	ID	D*	D*	D*	D*	D*	D*	A0	A1	A2	A3	MEM	WB		
MEM[R0+4] ← D3			IF														
MEM[R0+12] ← D0																	

- Is forwarding needed in the EX stage?
  - i2: yes: D0 is taken from pipeline register M6/MEM
  - i3:
  - i4:

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB						
D3 ← D0 + D5		IF	ID	D*	D*	D*	D*	D*	D*	A0	A1	A2	A3	MEM	WB		
MEM[R0+4] ← D3			IF	S*	S*	S*	S*	S*	S*								
MEM[R0+12] ← D0																	

- Is forwarding needed in the EX stage?
  - i2: yes: D0 is taken from pipeline register M6/MEM
  - i3:
  - i4:

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB						
D3 ← D0 + D5		IF	ID	D*	D*	D*	D*	D*	D*	A0	A1	A2	A3	MEM	WB		
MEM[R0+4] ← D3			IF	S*	S*	S*	S*	S*	S*	ID	EX						
MEM[R0+12] ← D0																	

- Is forwarding needed in the EX stage?
  - i2: yes: D0 is taken from pipeline register M6/MEM
  - i3:
  - i4:

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB						
D3 ← D0 + D5		IF	ID	D*	D*	D*	D*	D*	D*	A0	A1	A2	A3	MEM	WB		
MEM[R0+4] ← D3			IF	S*	S*	S*	S*	S*	S*	ID	EX	D*	D*				
MEM[R0+12] ← D0																	

- Is forwarding needed in the EX stage?
  - i2: yes: D0 is taken from pipeline register M6/MEM
  - i3:
  - i4:

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB						
D3 ← D0 + D5		IF	ID	D*	D*	D*	D*	D*	D*	A0	A1	A2	A3	MEM	WB		
MEM[R0+4] ← D3			IF	S*	S*	S*	S*	S*	S*	ID	EX	D*	D*	S*			
MEM[R0+12] ← D0																	

- Is forwarding needed in the EX stage?
  - i2: yes: D0 is taken from pipeline register M6/MEM
  - i3:
  - i4:

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB						
D3 ← D0 + D5		IF	ID	D*	D*	D*	D*	D*	D*	A0	A1	A2	A3	MEM	WB		
MEM[R0+4] ← D3			IF	S*	S*	S*	S*	S*	S*	ID	EX	D*	D*	S*	MEM	WB	
MEM[R0+12] ← D0																	

- Is forwarding needed in the EX stage?
  - i2: yes: D0 is taken from pipeline register M6/MEM
  - i3:
  - i4:

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB						
D3 ← D0 + D5		IF	ID	D*	D*	D*	D*	D*	D*	A0	A1	A2	A3	MEM	WB		
MEM[R0+4] ← D3			IF	S*	S*	S*	S*	S*	S*	ID	EX	D*	D*	S*	MEM	WB	
MEM[R0+12] ← D0																	

- Is forwarding needed in the EX stage?
  - i2: yes: D0 is taken from pipeline register M6/MEM
  - i3: yes: D3 is taken from pipeline register MEM/WB (or A3/MEM)
  - i4:

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB						
D3 ← D0 + D5		IF	ID	D*	D*	D*	D*	D*	D*	A0	A1	A2	A3	MEM	WB		
MEM[R0+4] ← D3			IF	S*	S*	S*	S*	S*	S*	ID	EX	D*	D*	S*	MEM	WB	
MEM[R0+12] ← D0										IF	ID						

- Is forwarding needed in the EX stage?
  - i2: yes: D0 is taken from pipeline register M6/MEM
  - i3: yes: D3 is taken from pipeline register MEM/WB (or A3/MEM)
  - i4:

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB						
D3 ← D0 + D5		IF	ID	D*	D*	D*	D*	D*	D*	A0	A1	A2	A3	MEM	WB		
MEM[R0+4] ← D3			IF	S*	S*	S*	S*	S*	S*	ID	EX	D*	D*	S*	MEM	WB	
MEM[R0+12] ← D0										IF	ID	S*	S*	S*			

- Is forwarding needed in the EX stage?
  - i2: yes: D0 is taken from pipeline register M6/MEM
  - i3: yes: D3 is taken from pipeline register MEM/WB (or A3/MEM)
  - i4:

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB						
D3 ← D0 + D5		IF	ID	D*	D*	D*	D*	D*	D*	A0	A1	A2	A3	MEM	WB		
MEM[R0+4] ← D3			IF	S*	S*	S*	S*	S*	S*	ID	EX	D*	D*	S*	MEM	WB	
MEM[R0+12] ← D0										IF	ID	S*	S*	S*	EX	MEM	WB

- Is forwarding needed in the EX stage?
  - i2: yes: D0 is taken from pipeline register M6/MEM
  - i3: yes: D3 is taken from pipeline register MEM/WB (or A3/MEM)
  - i4:

TASK 3 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
D0 ← D1 * D2	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB						
D3 ← D0 + D5		IF	ID	D*	D*	D*	D*	D*	D*	A0	A1	A2	A3	MEM	WB		
MEM[R0+4] ← D3			IF	S*	S*	S*	S*	S*	S*	ID	EX	D*	D*	S*	MEM	WB	
MEM[R0+12] ← D0										IF	ID	S*	S*	S*	EX	MEM	WB

- Is forwarding needed in the EX stage?
  - i2: yes: D0 is taken from pipeline register M6/MEM
  - i3: yes: D3 is taken from pipeline register MEM/WB (or A3/MEM)
  - i4: no

TASK 3 – SOLUTION

d) Reorder the instructions to improve the execution time (while keeping the semantics of the program)

- Observation: D0 is ready earlier than D3
- Let us write it into the memory first

Old code:

i1: D0 ← D1 * D2

i2: D3 ← D0 + D5

i3: MEM [R0+4] ← D3

i4: MEM [R0+12] ← D0

Optimized code:

D0 ← D1 * D2

D3 ← D0 + D5

MEM [R0+12] ← D0

MEM [R0+4] ← D3

TASK 4

- Consider the following sequence of instructions.

  i1: D2 ← D0 * D1

  i2: MEM [R0+0] ← D2

  i3: D2 ← D0 + D1

  i4: MEM [R0+8] ← D2

- The pipeline is the same as in the previous example.

  a) Identify all the data dependencies among the instructions

  b) Schedule the instructions in the pipeline

TASK 4 – SOLUTION

- Consider the following sequence of instructions.

  i1: **D2** ← D0 * D1

  i2: MEM [R0+0] ← D2

  i3: D2 ← D0 + D1

  i4: MEM [R0+8] ← D2

- The pipeline is the same as in the previous example.

  a) Identify all the data dependencies among the instructions

  RAW:

TASK 4 – SOLUTION

- Consider the following sequence of instructions.

  i1: **D2** ← D0 * D1

  i2: MEM [R0+0] ← **D2**

  i3: D2 ← D0 + D1

  i4: MEM [R0+8] ← D2

- The pipeline is the same as in the previous example.

  a) Identify all the data dependencies among the instructions

  RAW: i1 ↔ i2,

TASK 4 – SOLUTION

- Consider the following sequence of instructions.

  i1: **D2** ← D0 * D1

  i2: MEM [R0+0] ← D2

  i3: D2 ← D0 + D1

  i4: MEM [R0+8] ← **D2**

- The pipeline is the same as in the previous example.

  a) Identify all the data dependencies among the instructions

  RAW: i1 ↔ i2, (i1 ↔ i4)

TASK 4 – SOLUTION

- Consider the following sequence of instructions.

  i1: D2 ← D0 * D1

  i2: MEM [R0+0] ← D2

  i3: **D2** ← D0 + D1

  i4: MEM [R0+8] ← **D2**

- The pipeline is the same as in the previous example.

  a) Identify all the data dependencies among the instructions

  RAW: i1 ↔ i2, ~~(i1 ↔ i4)~~, i3 ↔ i4

TASK 4 – SOLUTION

- Consider the following sequence of instructions.

  i1: **D2** ← D0 * D1

  i2: MEM [R0+0] ← D2

  i3: **D2** ← D0 + D1

  i4: MEM [R0+8] ← D2

- The pipeline is the same as in the previous example.

  a) Identify all the data dependencies among the instructions

  RAW: i1 ↔ i2, ~~(i1 ↔ i4)~~, i3 ↔ i4

  WAW: i1 ↔ i3

TASK 4 – SOLUTION

- Consider the following sequence of instructions.

  i1: D2 ← D0 * D1

  i2: MEM [R0+0] ← **D2**

  i3: **D2** ← D0 + D1

  i4: MEM [R0+8] ← D2

- The pipeline is the same as in the previous example.

  a) Identify all the data dependencies among the instructions

  RAW: i1 ↔ i2, ~~(i1 ↔ i4)~~, i3 ↔ i4

  WAW: i1 ↔ i3

  WAR: i2 ↔ i3

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1														
MEM[R0+0] ← D2														
D2 ← D0 + D1														
MEM[R0+8] ← D2														

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1	IF	ID												
MEM[R0+0] ← D2														
D2 ← D0 + D1														
MEM[R0+8] ← D2														

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1	IF	ID	M0	M1	M2	M3	M4	M5	M6					
MEM[R0+0] ← D2														
D2 ← D0 + D1														
MEM[R0+8] ← D2														

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB			
MEM[R0+0] ← D2														
D2 ← D0 + D1														
MEM[R0+8] ← D2														

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB			
MEM[R0+0] ← D2		IF	ID	EX										
D2 ← D0 + D1														
MEM[R0+8] ← D2														

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB			
MEM[R0+0] ← D2		IF	ID	EX	D*	D*	D*	D*	D*					
D2 ← D0 + D1														
MEM[R0+8] ← D2														

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB			
MEM[R0+0] ← D2		IF	ID	EX	D*	D*	D*	D*	D*	S*				
D2 ← D0 + D1														
MEM[R0+8] ← D2														

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB			
MEM[R0+0] ← D2		IF	ID	EX	D*	D*	D*	D*	D*	S*	MEM	WB		
D2 ← D0 + D1														
MEM[R0+8] ← D2														

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB			
MEM[R0+0] ← D2		IF	ID	EX	D*	D*	D*	D*	D*	S*	MEM	WB		
D2 ← D0 + D1			IF	ID										
MEM[R0+8] ← D2														

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB			
MEM[R0+0] ← D2		IF	ID	EX	D*	D*	D*	D*	D*	S*	MEM	WB		
D2 ← D0 + D1			IF	ID	A0	A1	A2	A3						
MEM[R0+8] ← D2														

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB			
MEM[R0+0] ← D2		IF	ID	EX	D*	D*	D*	D*	D*	S*	MEM	WB		
D2 ← D0 + D1			IF	ID	A0	A1	A2	A3	MEM					
MEM[R0+8] ← D2														

- May i3 enter into the MEM stage in cycle 9?
  - Is MEM stage free?
  - Can we continue execution after entering into MEM?

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB			
MEM[R0+0] ← D2		IF	ID	EX	D*	D*	D*	D*	D*	S*	MEM	WB		
D2 ← D0 + D1			IF	ID	A0	A1	A2	A3	MEM					
MEM[R0+8] ← D2														

- May i3 enter into the MEM stage in cycle 9?
  - Is MEM stage free?
  - Can we continue execution after entering into MEM?
  - Please recall: WAW: i1 ↔ i3

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB			
MEM[R0+0] ← D2		IF	ID	EX	D*	D*	D*	D*	D*	S*	MEM	WB		
D2 ← D0 + D1			IF	ID	A0	A1	A2	A3	D*/S*					
MEM[R0+8] ← D2														

- Comment:
  - In cycle 9, i3 must wait due to WAW: i1 ← i3:
    - i3 may not write its result into D2 before i1
    - entering into MEM would result in a deadlock

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB			
MEM[R0+0] ← D2		IF	ID	EX	D*	D*	D*	D*	D*	S*	MEM	WB		
D2 ← D0 + D1			IF	ID	A0	A1	A2	A3	D*/S*	S*	S*			
MEM[R0+8] ← D2														

- Comment:
  - In cycle 9, i3 must wait due to WAW: i1 ← i3:
    - i3 may not write its result into D2 before i1
    - entering into MEM would result in a deadlock
    - i3 may enter into MEM only in cycle 12

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB			
MEM[R0+0] ← D2		IF	ID	EX	D*	D*	D*	D*	D*	S*	MEM	WB		
D2 ← D0 + D1			IF	ID	A0	A1	A2	A3	D*/S*	S*	S*	MEM	WB	
MEM[R0+8] ← D2														

- Comment:
  - In cycle 9, i3 must wait due to WAW: i1 ← i3:
    - i3 may not write its result into D2 before i1
    - entering into MEM would result in a deadlock
    - i3 may enter into MEM only in cycle 12

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB			
MEM[R0+0] ← D2		IF	ID	EX	D*	D*	D*	D*	D*	S*	MEM	WB		
D2 ← D0 + D1			IF	ID	A0	A1	A2	A3	D*/S*	S*	S*	MEM	WB	
MEM[R0+8] ← D2				IF	ID									

- Comment:
  - In cycle 9, i3 must wait due to WAW: i1 ← i3:
    - i3 may not write its result into D2 before i1
    - entering into MEM would result in a deadlock
    - i3 may enter into MEM only in cycle 12

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB			
MEM[R0+0] ← D2		IF	ID	EX	D*	D*	D*	D*	D*	S*	MEM	WB		
D2 ← D0 + D1			IF	ID	A0	A1	A2	A3	D*/S*	S*	S*	MEM	WB	
MEM[R0+8] ← D2				IF	ID	S*	S*	S*	S*	S*				

- Comment:
  - In cycle 9, i3 must wait due to WAW: i1 ← i3:
    - i3 may not write its result into D2 before i1
    - entering into MEM would result in a deadlock
    - i3 may enter into MEM only in cycle 12

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB			
MEM[R0+0] ← D2		IF	ID	EX	D*	D*	D*	D*	D*	S*	MEM	WB		
D2 ← D0 + D1			IF	ID	A0	A1	A2	A3	D*/S*	S*	S*	MEM	WB	
MEM[R0+8] ← D2				IF	ID	S*	S*	S*	S*	S*	EX			

- Comment:
  - In cycle 9, i3 must wait due to WAW: i1 ← i3:
    - i3 may not write its result into D2 before i1
    - entering into MEM would result in a deadlock
    - i3 may enter into MEM only in cycle 12

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB			
MEM[R0+0] ← D2		IF	ID	EX	D*	D*	D*	D*	D*	S*	MEM	WB		
D2 ← D0 + D1			IF	ID	A0	A1	A2	A3	D*/S*	S*	S*	MEM	WB	
MEM[R0+8] ← D2				IF	ID	S*	S*	S*	S*	S*	EX	S*		

- Comment:
  - In cycle 9, i3 must wait due to WAW: i1 ← i3:
    - i3 may not write its result into D2 before i1
    - entering into MEM would result in a deadlock
    - i3 may enter into MEM only in cycle 12

TASK 4 – SOLUTION

Instructions:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← D0 * D1	IF	ID	M0	M1	M2	M3	M4	M5	M6	MEM	WB			
MEM[R0+0] ← D2		IF	ID	EX	D*	D*	D*	D*	D*	S*	MEM	WB		
D2 ← D0 + D1			IF	ID	A0	A1	A2	A3	D*/S*	S*	S*	MEM	WB	
MEM[R0+8] ← D2				IF	ID	S*	S*	S*	S*	S*	EX	S*	MEM	WB

- Comment:
  - In cycle 9, i3 must wait due to WAW: i1 ← i3:
    - i3 may not write its result into D2 before i1
    - entering into MEM would result in a deadlock
    - i3 may enter into MEM only in cycle 12