

A.5

a)

IF	needs an adder for PC
RF, WB	do not need an adder
ALU1	needs an adder to compute effective address of some memory instructions
MEM	does not need adder.
ALU	may use an adder for ALU operations.

In the worst case you would need 3 adders. One for each IF, ALU1, ALU2 stage consider:

```
add    r1,r2,r3
nop
add    r4,r0,0(r7)
nop
nop
```

IF would process any instruction, ALU1 has mem address computation ALU2 needs adder to compute results.

b)

IF, ALU1, MEM, ALU2 do not access the register file thus they don't need register r/w ports.
RF needs read ports for Rsrc1 and Rsrc2 values.
WB needs 1 write port to store values into Rdest.

In the worst case this architecture needs 2 read ports and 1 write port for its register file. consider:

```
add    r1,r2,r3
stall
stall
stall
add    r4,r5,r6
```

Furthermore we need 1 r/w memory port. 1 memory read for IF, potentially you need an extra write if MEM requires access to memory.

c)

- ALU1 stage computer the address of a MEM access. Its result provides the address of the memory access requires by the instruction to the MEM stage of the pipeline.

-ALU2 stage performs any ALU operations required by an instruction. For instance an ADD will perform its addition at this stage.

SOURCE	INSTRUCTION	DESTINATION	INSTRUCTION	
Pipeline Reg	OPcode	Pipeline Reg	OPcode	Dest. Result
ALU2/WB	ALU OP	MEM/ALU2	ALU OP	ALU2 Input
ALU2/WB	ALU OP	ALU1/MEM	ALU OP	Source
ALU2/WB	ALU OP	RF/ALU1	ALU OP	Source
ALU2/WB	ALU OP	RF/ALU1	EA OP	ALU1 Input

A path to Source 1 input that is activated when the destination reg. of the source instr. is the same as the source 1 reg. of the destination instr.

A path to the source 2 input that is activated when the destination register of the source instruction is the same as the source 2 register of the destination instr.

d)

SOURCE	INSTRUCTION	DESTINATION	INSTRUCTION	
Pipeline Reg	OPcode	Pipeline Reg	OPcode	Dest. Result
MEM/ALU2	Load	RF/ALU1	EA Op	ALU1 Input
ALU2/WB	Load	RF/ALU1	EA Op	ALU1 Input
ALU2/WB	Load	MEM/ALU2	ALU OP	ALU2 Input
ALU2/WB	Load	ALU1/MEM	ALU OP	Source
ALU2/WB	Load	RF/ALU1	ALU OP	Source
MEM/ALU2	Load	ALU1/MEM	Store	Source
MEM/ALU2	Load	RF/ALU1	Store	Source
ALU2/WB	Load	RF/ALU1	Store	Source
ALU2/WB	ALU OP	ALU1/MEM	Store	Source
ALU2/WB	ALU OP	RF/ALU1	Store	Source

Destination of the result can either be an ALU input or a source used to carry the value of the src. reg. down the pipeline. ALU OP is any instruction that uses ALU2 to do computation. EA Op is instruction that computes effective address.

Two forwarding paths:

Path to Source1 input when the destination register of the source instruction is the same as the source 1 register of the destination instruction.

Path to Source 2 input when the destination register of the source instruction is the same as the source 2 register of the destination instruction.

e)

Opcode #1 In IF/RF Register	Opcode #2	Pipeline Register	Latency
EA Op	Load	RF/ALU1	1
EA Op	ALU Op	RF/ALU1	2
Store	ALU Op	RF/ALU1	1
EA Op	ALU Op	ALU1/MEM	1

f)

Instruction	Clock Cycle							
	1	2	3	4	5	6	7	8
Branch	IF	RF	ALU1	MEM	ALU2	WB		
Branch + 1		IF	stall	stall	stall	IF	RF	ALU1
Branch + 2							IF	RF

A.6

a)

We can modify the traditional five stage pipeline of IF, ID, EX, MEM, WB to IF, ID, MEM, EX, WB if we are to support register indirect addressing.

b)

We only require EX to WB forwarding.

c)

Source Instruction Pipeline Stage	Destination Instruction Pipeline Stage	
MEM	MEM	Load or Store (data value or index)
EX	MEM	ALU OP register to Load / Store
EX	EX	ALU Dependency

d)

The modified pipeline enables the the merging of loads and ALU operations, thus reducing instruction count. But, we cannot merge them if the memory operation requires an offset. Thus the modified ISA requires an additional instruction to compute the offset value and store it in a register.

Old ISA	New ISA	
LD R2, 16(R1)	ADDUI R3, R1, #16 LD R2, (R3)	0 new instructions
LD R2, 0(R1) ADD R2, R3, R4	ADD R3, R4, (R1)	2in1

e)

The old pipeline was producing stalls on loads producing a value for the next instruction. In the new pipeline this isn't the case, thus the CPI will decrease. An offset however that is needed by a MEM instruction will cause 1 stall in the execute stage.

	Old ISA	New ISA
LD R6, 0(R1) ADD R7, R6, R8	1 stall	0 stalls
ADDUI R6, R5, #8 LD R7, 0(R6)	0 stalls	1 stall