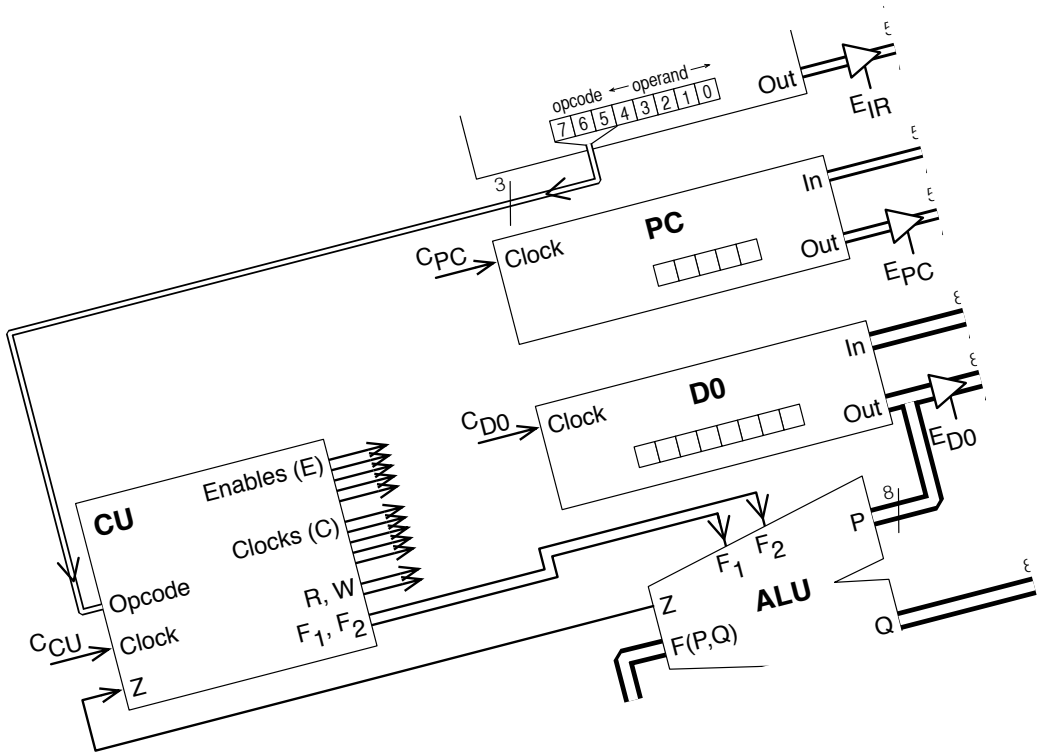
CS132 T6: Processor Architectures



- Two single-sheet handouts (H1 and H2) today

T6 overview

- Building a CPU containing
  - Arithmetic Logic Unit
  - Registers
  - Buses
  - Control Unit
- Implementing a Control Unit
- CISC vs RISC

T6 learning outcomes

- Understand distinction between "architecture" and "organisation"
- Understand the roles of the key components within the processor, and how these are used during the execution of software
- Understand the functionality of the control unit
- Be familiar with various ways of implementing a control unit
- Compare CISC and RISC design strategies

Lecture 1: Session outline

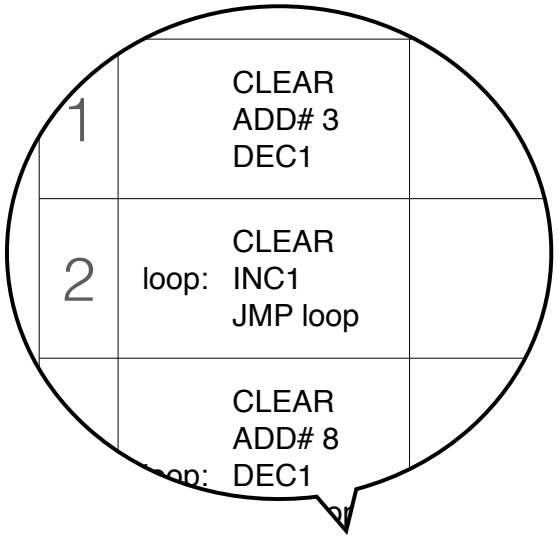
- Organisation vs Architecture
- Programming PATP
- PATP internal organisation
- Controlling PATP's components
- Questions

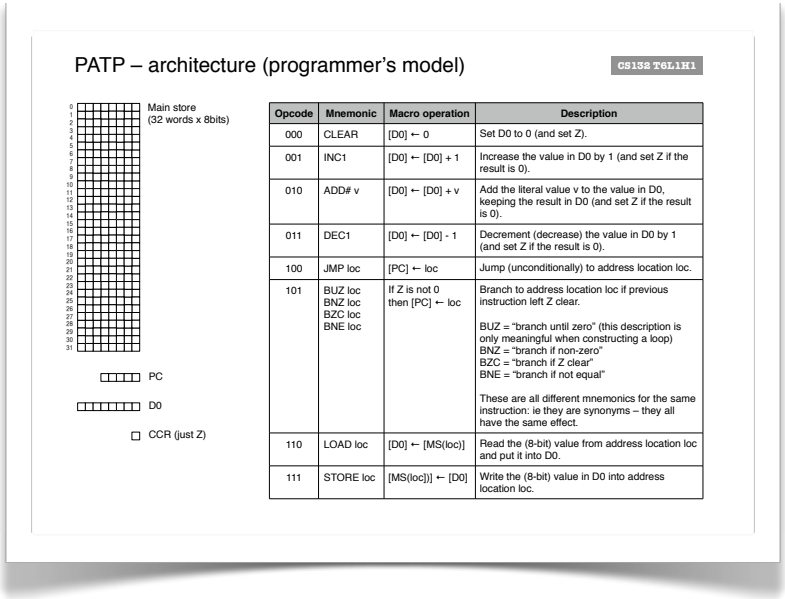
Organisation vs Architecture

Computer Architecture: structure & properties, as viewed from perspective of a software engineer

Computer Organisation: internal structure & properties, as viewed by a hardware engineer

Organisation vs Architecture





PATP – architecture (programmer's model)

CS132 T6L1H1

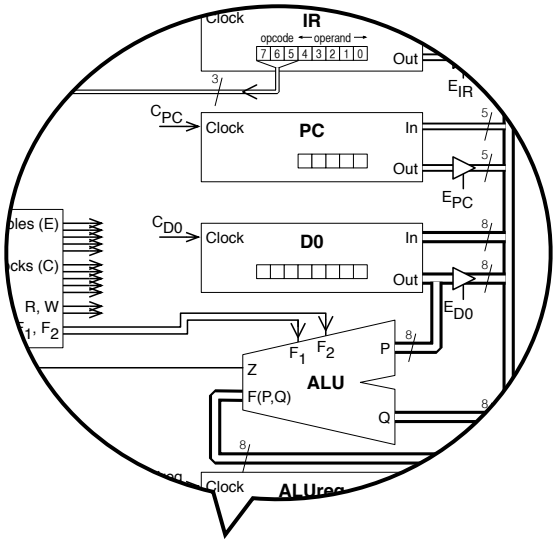
Main store (32 words x 8bits)

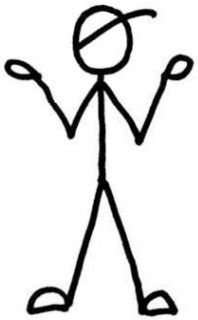
PC

D0

CCR (just Z)

Opcode	Mnemonic	Macro operation	Description
000	CLEAR	[D0] ← 0	Set D0 to 0 (and set Z).
001	INC1	[D0] ← [D0] + 1	Increase the value in D0 by 1 (and set Z if the result is 0).
010	ADD# v	[D0] ← [D0] + v	Add the literal value v to the value in D0, keeping the result in D0 (and set Z if the result is 0).
011	DEC1	[D0] ← [D0] - 1	Decrement (decrease) the value in D0 by 1 (and set Z if the result is 0).
100	JMP loc	[PC] ← loc	Jump (unconditionally) to address location loc.
101	BUZ loc BNZ loc BZC loc BNE loc	If Z is not 0 then [PC] ← loc	Branch to address location loc if previous instruction left Z clear. BUZ = "branch until zero" (this description is only meaningful when constructing a loop) BNZ = "branch if non-zero" BZC = "branch if Z clear" BNE = "branch if not equal" These are all different mnemonics for the same instruction: ie they are synonyms – they all have the same effect.
110	LOAD loc	[D0] ← [MS(loc)]	Read the (8-bit) value from address location loc and put it into D0.
111	STORE loc	[MS(loc)] ← [D0]	Write the (8-bit) value in D0 into address location loc.





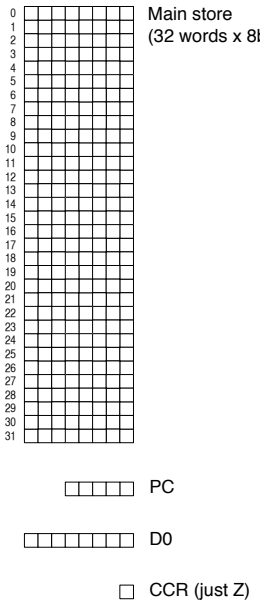
Software engineer

Hardware engineer

H1: architecture

PATP – architecture (programmer's model)

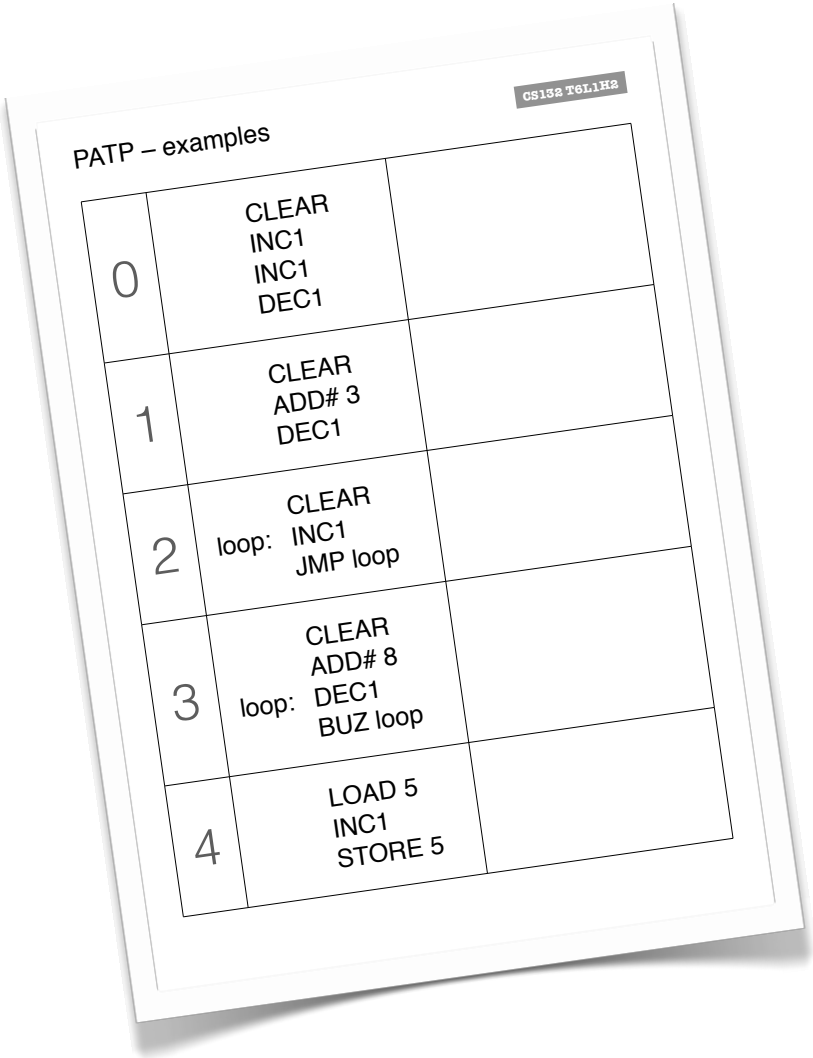
CS132 T6L1H1



Opcode	Mnemonic	Macro operation	Description
000	CLEAR	[D0] ← 0	Set D0 to 0 (and set Z).
001	INC1	[D0] ← [D0] + 1	Increase the value in D0 by 1 (and set Z if the result is 0).
010	ADD# v	[D0] ← [D0] + v	Add the literal value v to the value in D0, keeping the result in D0 (and set Z if the result is 0).
011	DEC1	[D0] ← [D0] - 1	Decrement (decrease) the value in D0 by 1 (and set Z if the result is 0).
100	JMP loc	[PC] ← loc	Jump (unconditionally) to address location loc.
101	BUZ loc BNZ loc BZC loc BNE loc	If Z is not 0 then [PC] ← loc	Branch to address location loc if previous instruction left Z clear. BUZ = "branch until zero" (this description is only meaningful when constructing a loop) BNZ = "branch if non-zero" BZC = "branch if Z clear" BNE = "branch if not equal" These are all different mnemonics for the same instruction: ie they are synonyms – they all have the same effect.
110	LOAD loc	[D0] ← [MS(loc)]	Read the (8-bit) value from address location loc and put it into D0.
111	STORE loc	[MS(loc])] ← [D0]	Write the (8-bit) value in D0 into address location loc.

PATP = "Pedagogically Adventurous Teaching Processor"

H2: examples



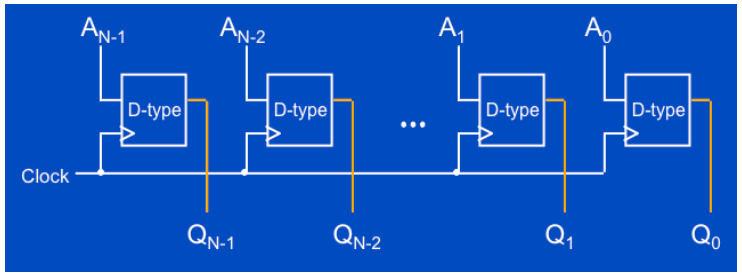
CS132 T6L1H2

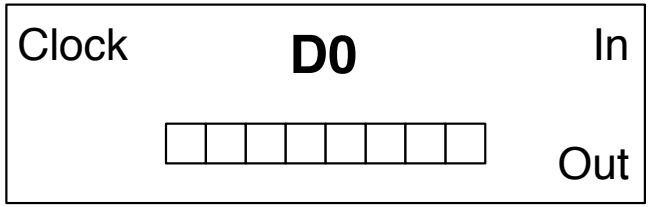
PATP – examples

0	CLEAR INC1 INC1 DEC1	
1	CLEAR ADD# 3 DEC1	
2	loop: CLEAR INC1 JMP loop	
3	loop: CLEAR ADD# 8 DEC1 BUZ loop	
4	LOAD 5 INC1 STORE 5	

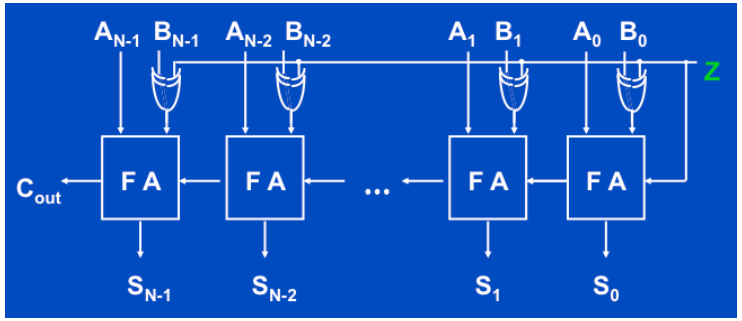
Building blocks

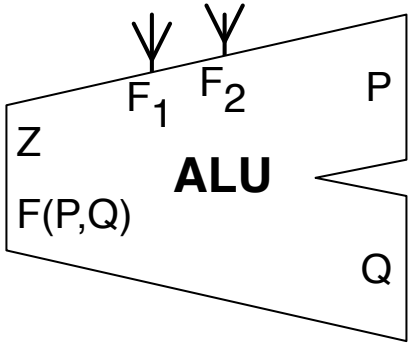
Register





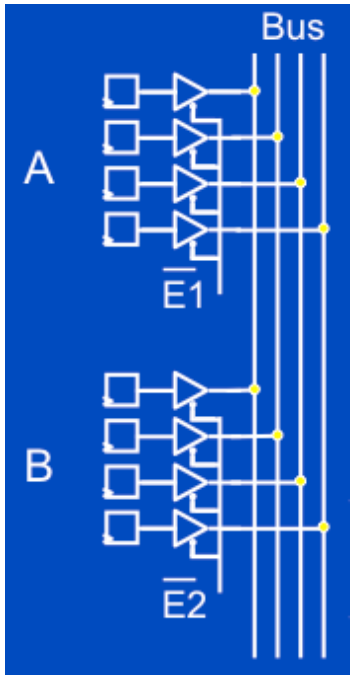
Arithmetic

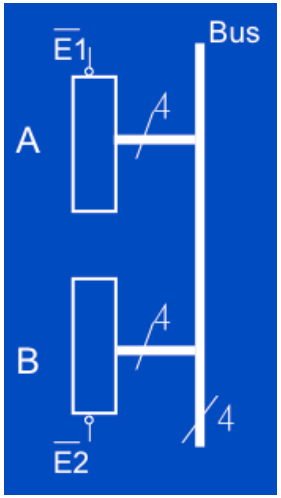


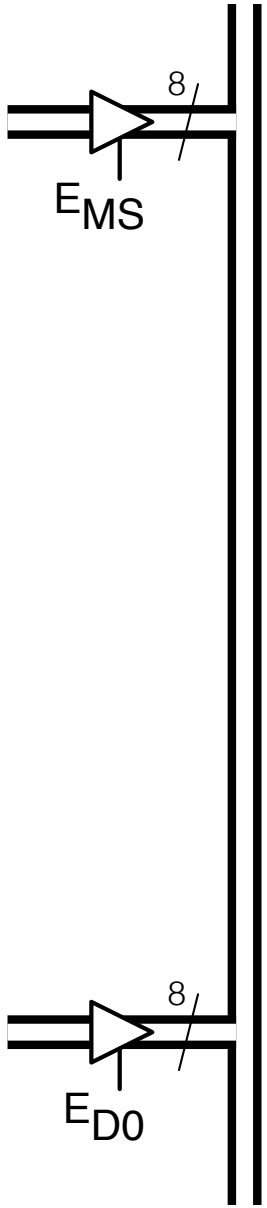


Building blocks

Bus & tri-state buffer

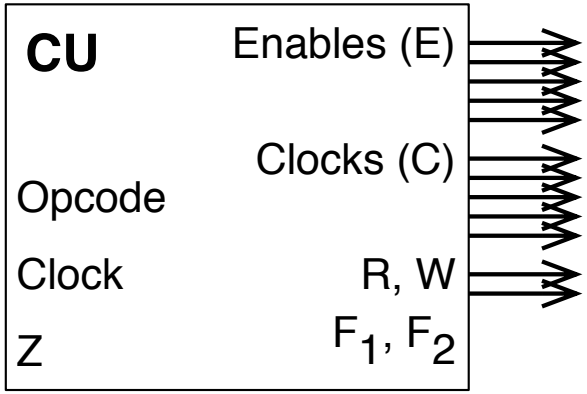






Building blocks

Control Unit



H3: internal organisation

PATP – internal organisation

CS132 T6L1H3

CU inputs

Opcode

C_{CU}

Z

CU outputs

C_{MAR}

E_{MS}

E_{IR} C_{IR}

E_{PC} C_{PC}

E_{D0} C_{D0}

E_{ALUreg} C_{ALUreg}

F_1

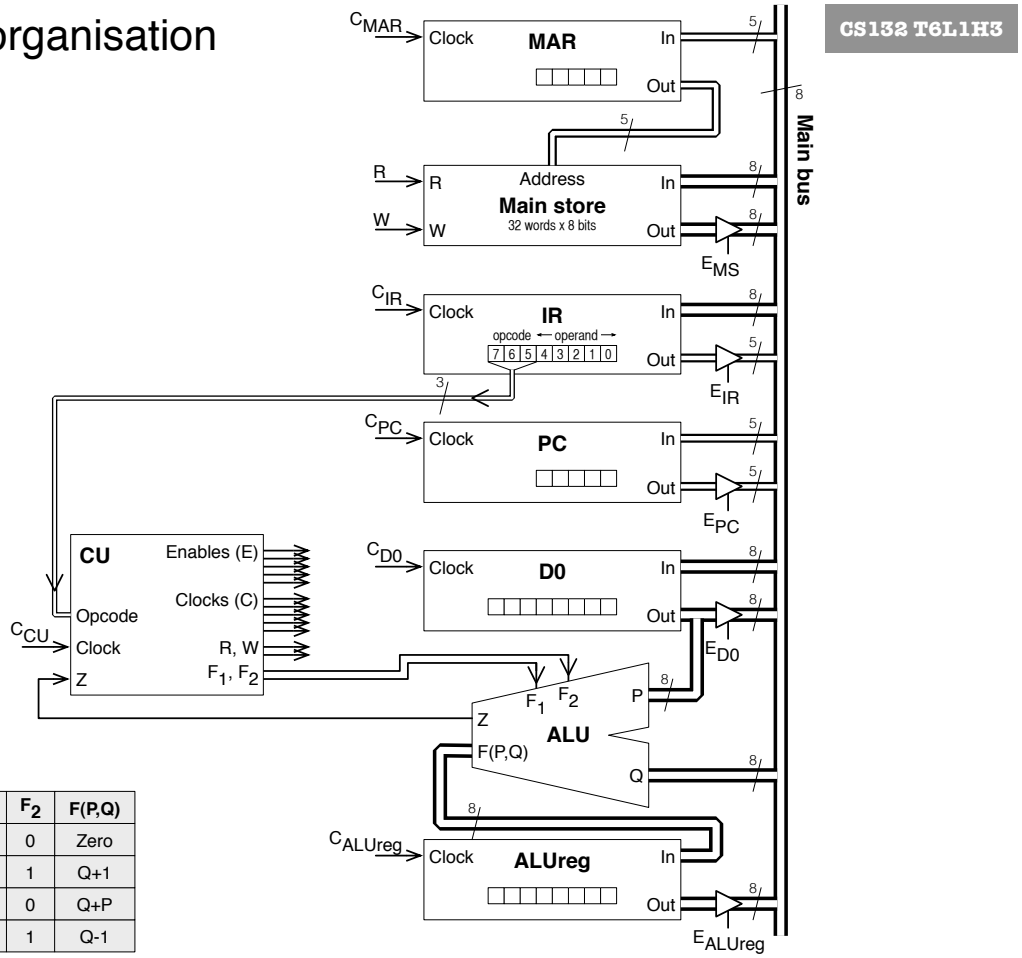
F_2

R

W

Opcode	Mnemonic
000	CLEAR
001	INC1
010	ADD# v
011	DEC1
100	JMP loc
101	BUZ loc (or BNZ, BZC, BNE)
110	LOAD loc
111	STORE loc

F_1	F_2	F(P,Q)
0	0	Zero
0	1	Q+1
1	0	Q+P
1	1	Q-1



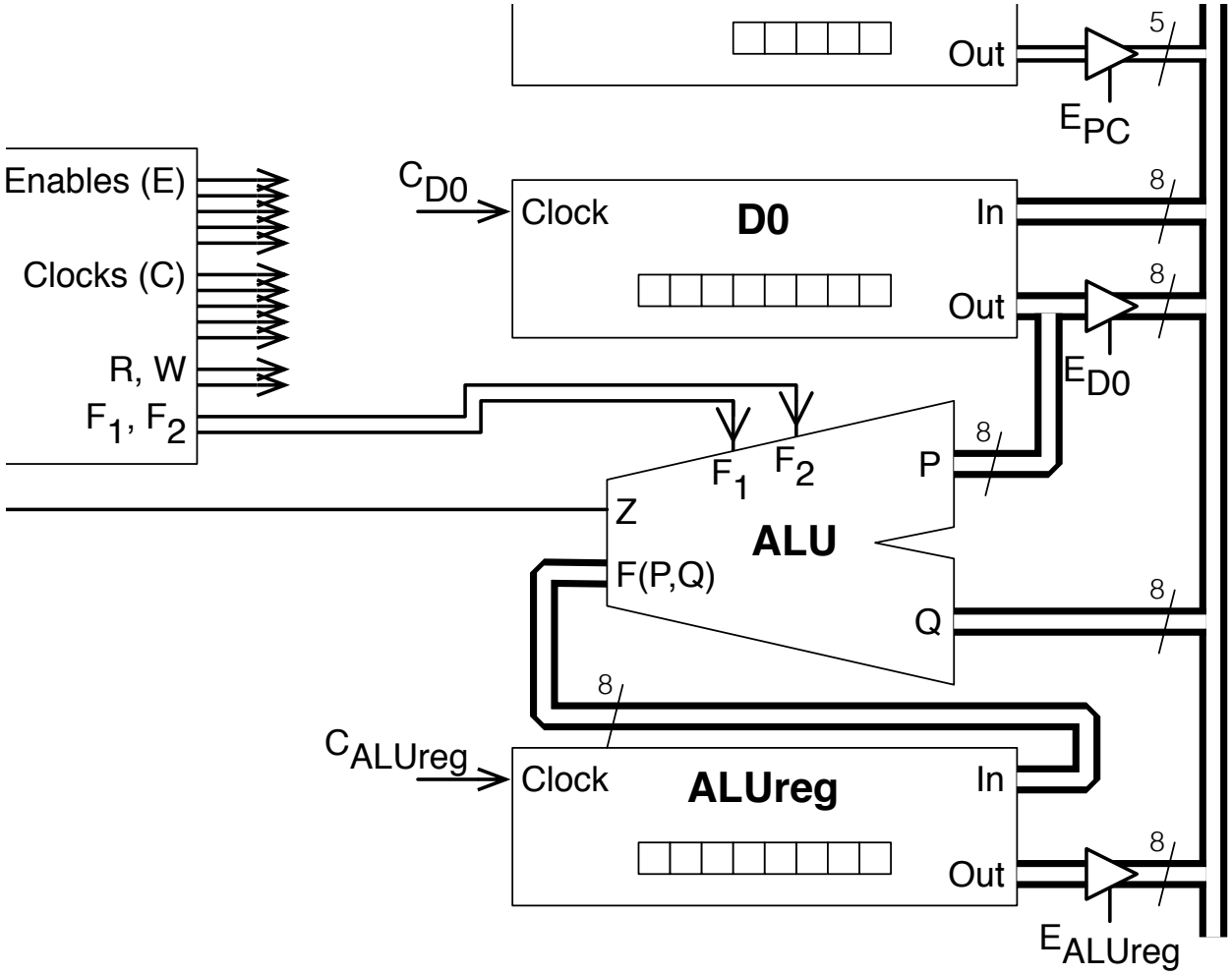
What control signals,
in what order,
would you need to assert,
in order to implement the INC1 instruction?

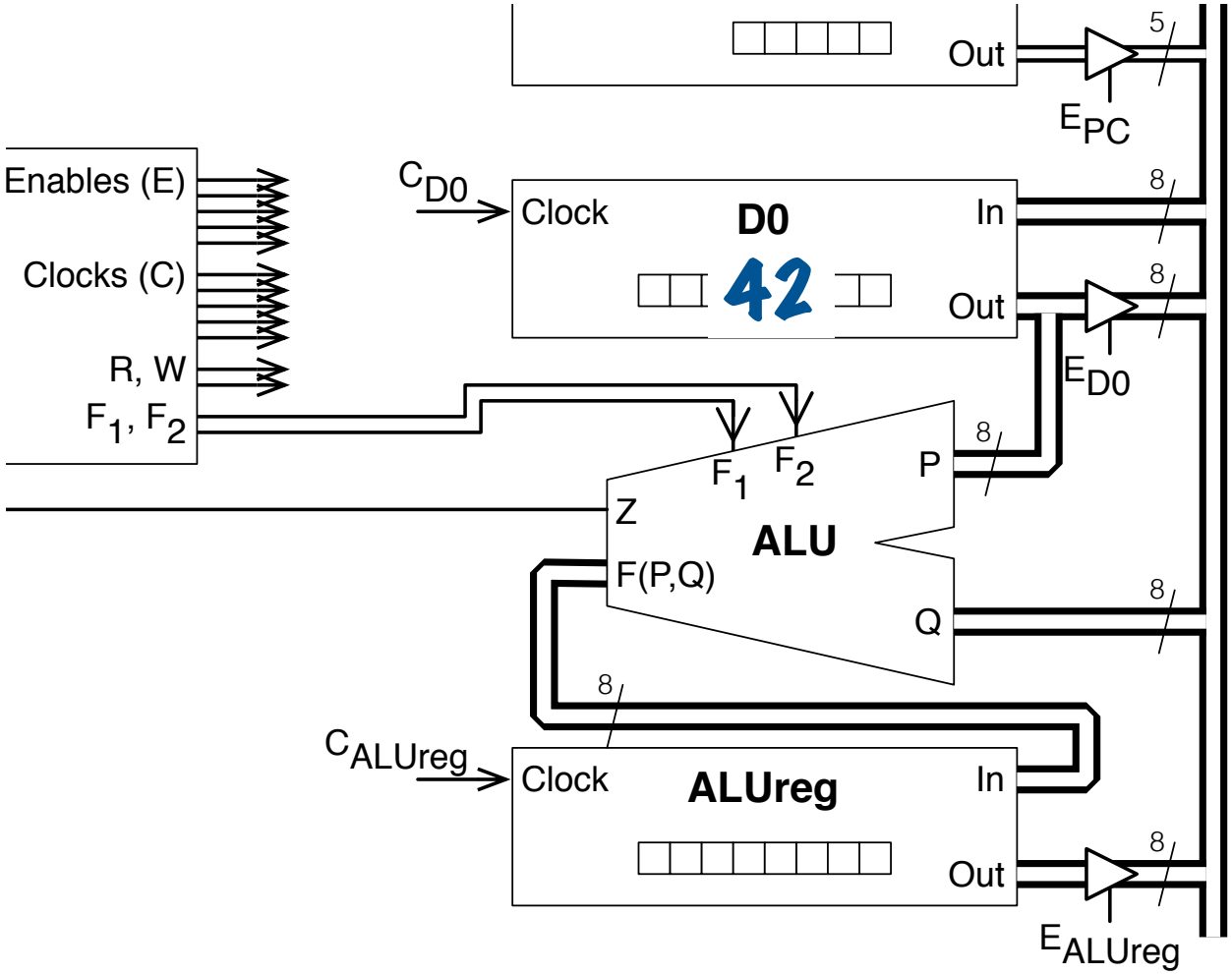
Hint: use the "Q+1" ALU function.

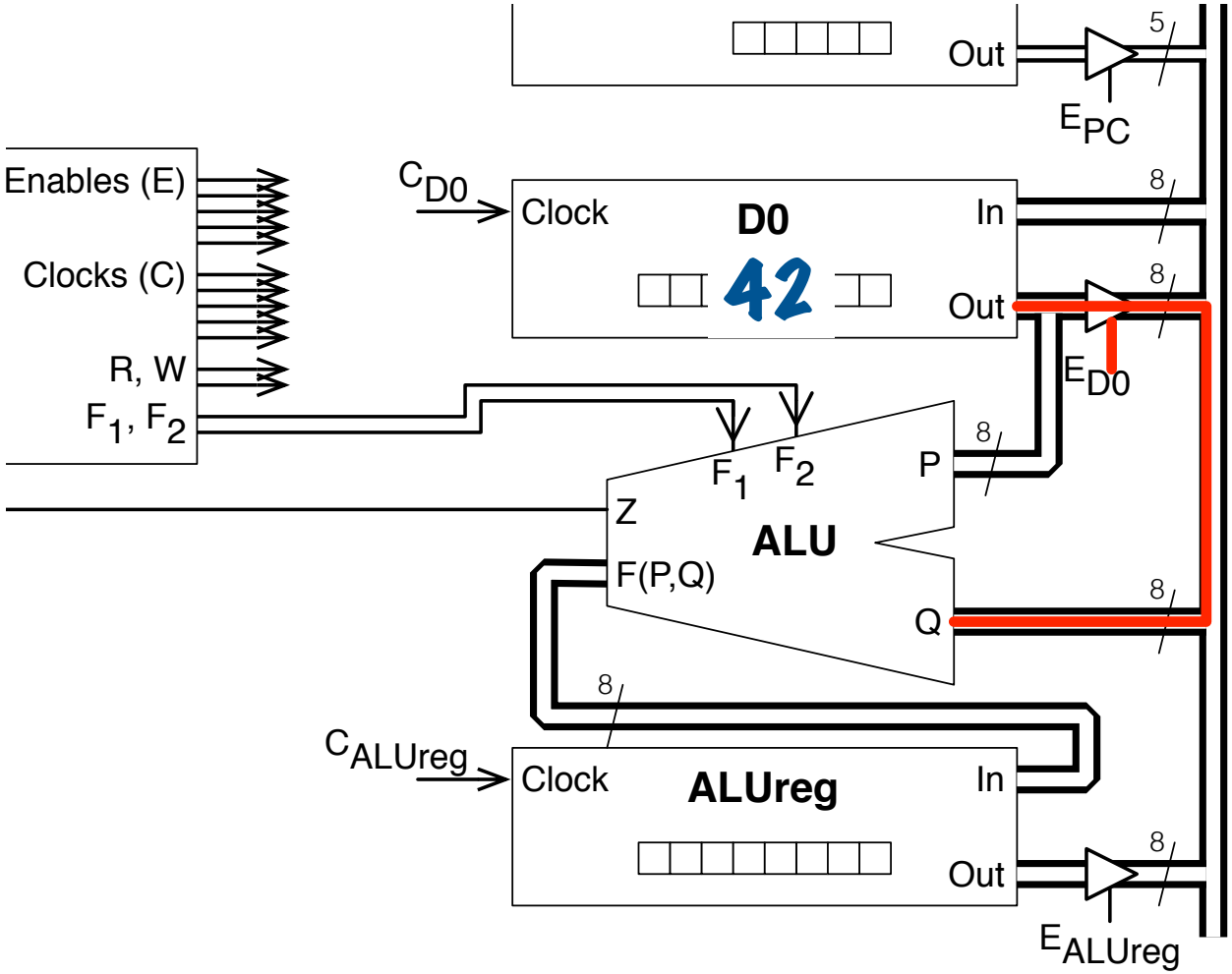
What control signals,
in what order,
would you need to assert,
in order to implement the INC1 instruction?

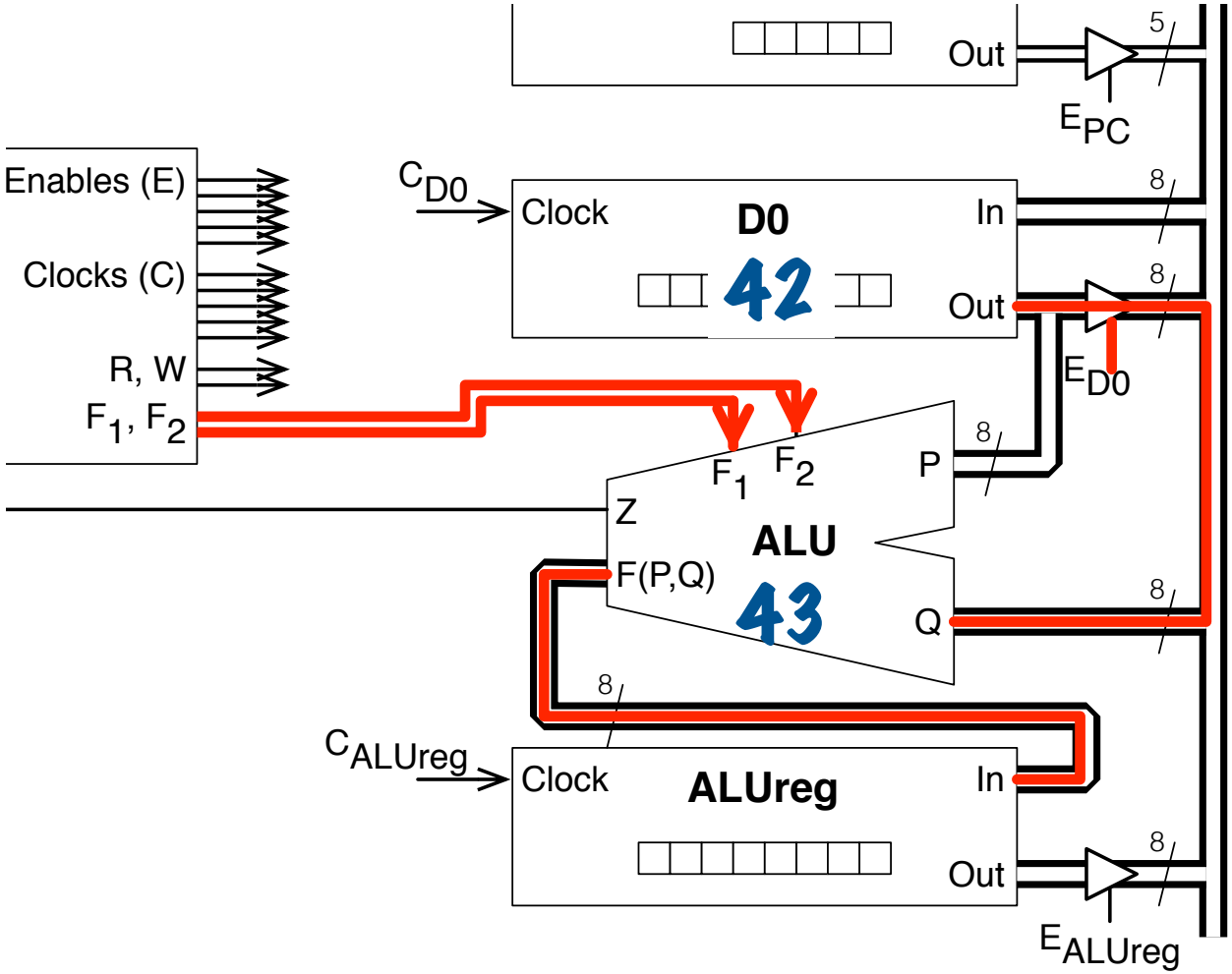
Hint: use the "Q+1" ALU function.

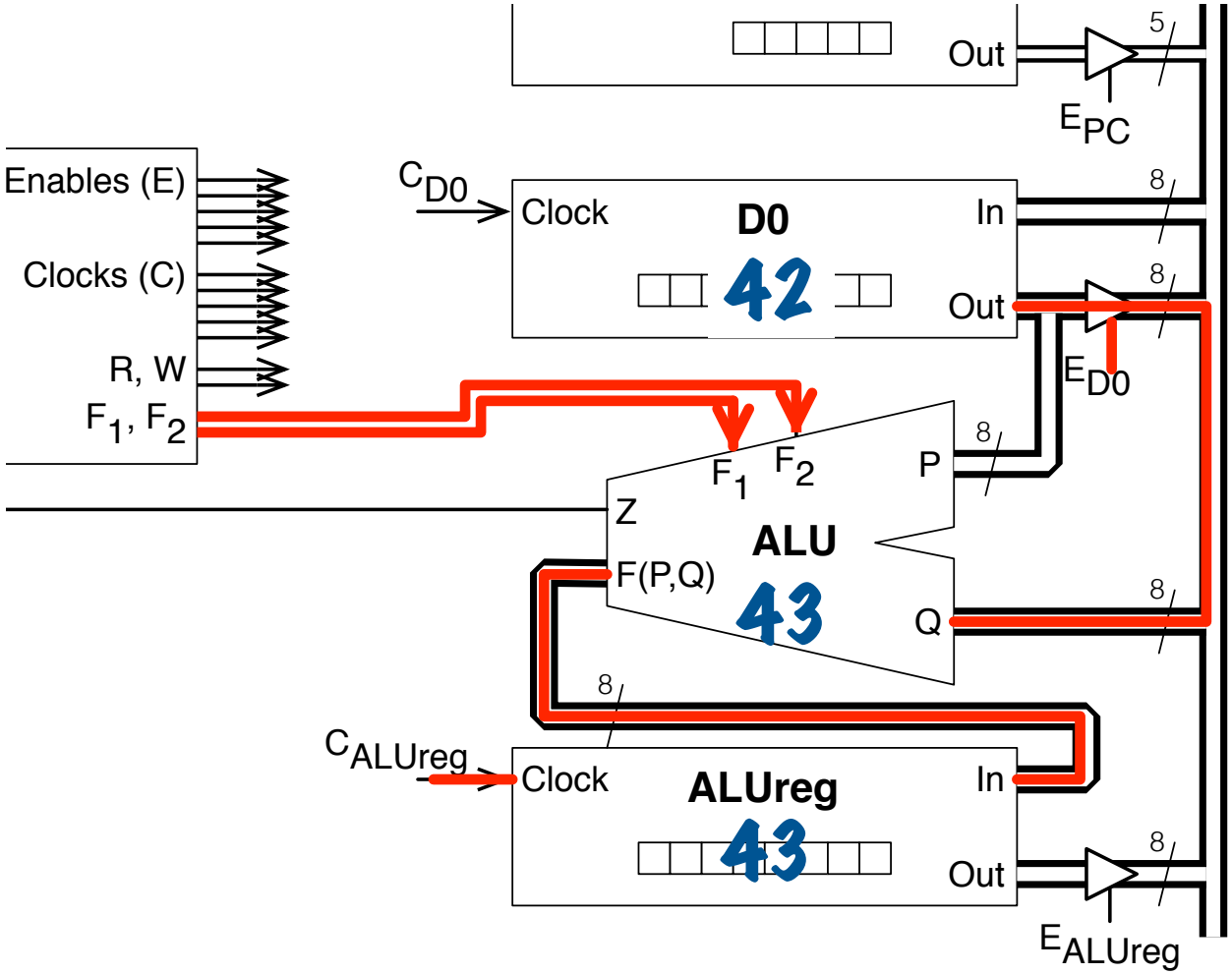
H4 gives operations required to implement all of example 0.
INC1 is micro steps 5-8.

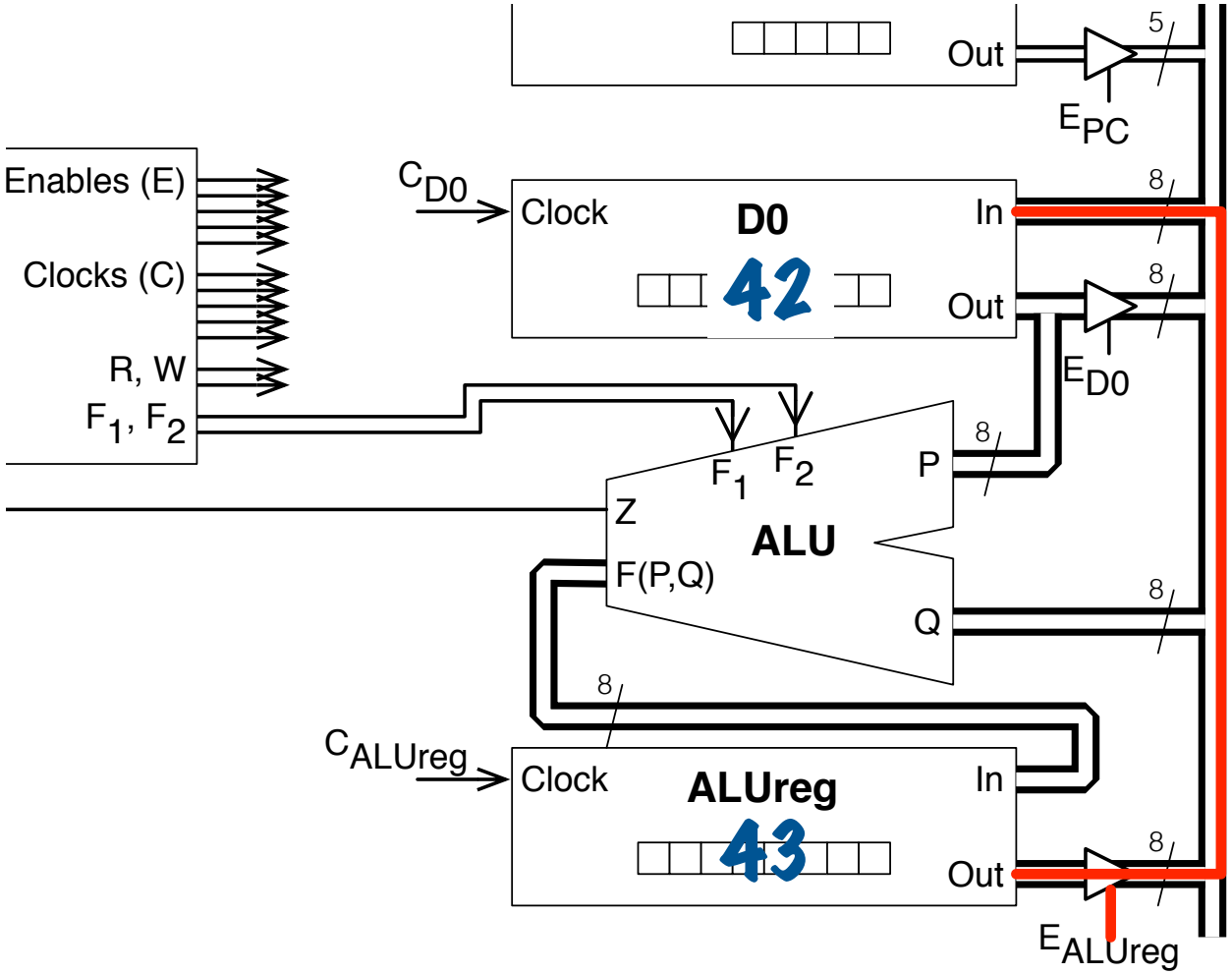

Out
5
E_{PC}
Enables (E)
C_{D0}
Clock
D0
In
8
Clocks (C)
8
Out
R, W
E_{D0}
F_1, F_2
8
F_1
F_2
P
Z
ALU
F(P,Q)
8
Q
8
C_{ALUreg}
Clock
ALUreg
In
8
Out
E_{ALUreg}

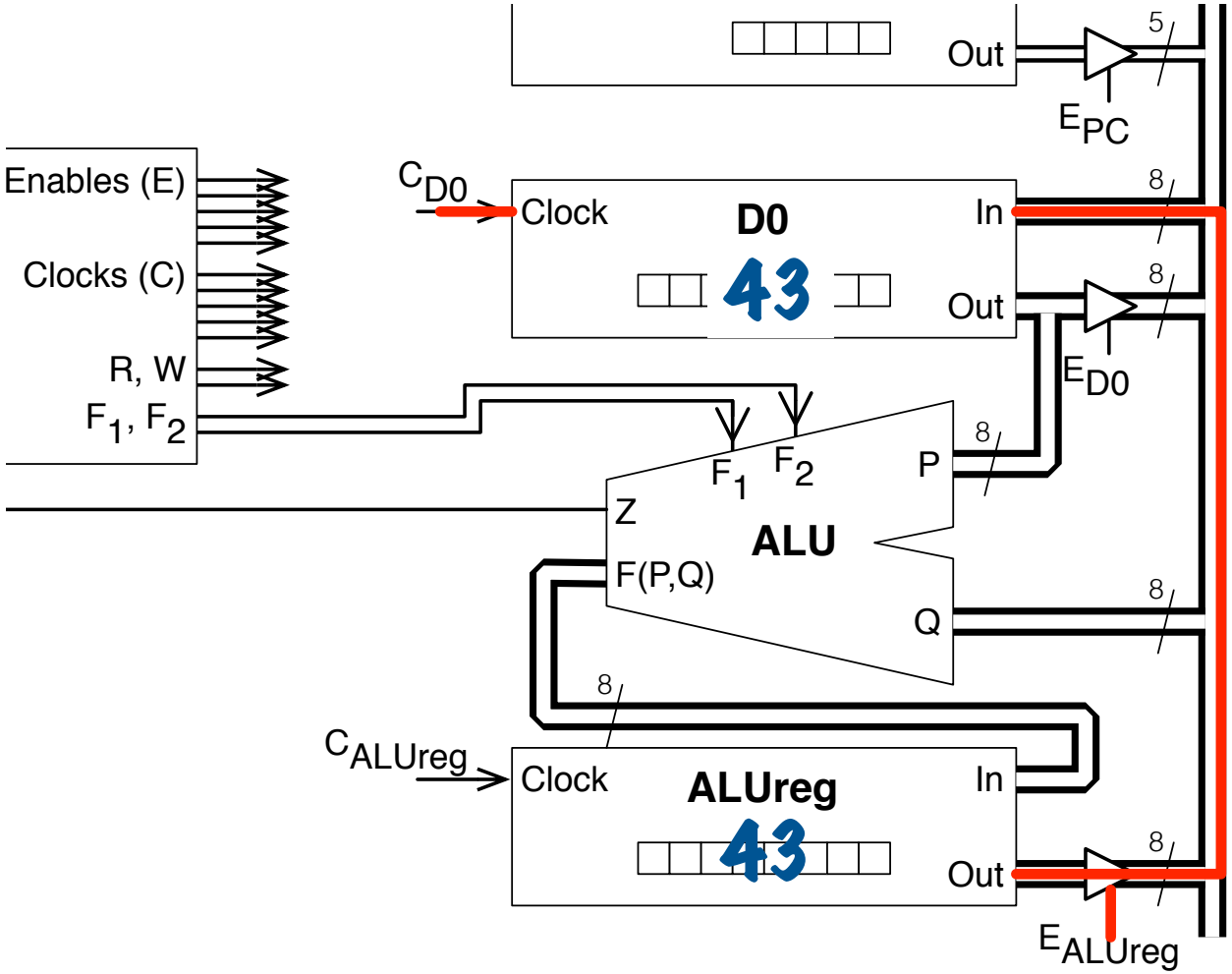

Out
5
E_{PC}
Enables (E)
C_{D0}
Clock
D0
In
8
Clocks (C)
42
Out
8
R, W
E_{D0}
F_1, F_2
8
F_1
F_2
P
Z
ALU
F(P,Q)
8
Q
8
C_{ALUreg}
Clock
ALUreg
In
8
Out
E_{ALUreg}


Out
5
E_{PC}
Enables (E)
C_{D0}
Clock
D0
In
8
Clocks (C)
42
Out
8
R, W
E_{D0}
F_1, F_2
F_1
F_2
P
8
Z
ALU
F(P,Q)
8
Q
8
C_{ALUreg}
Clock
ALUreg
In
8
Out
E_{ALUreg}


Out
5
E_{PC}
Enables (E)
C_{D0}
Clock
D0
In
8
Clocks (C)
42
Out
8
R, W
E_{D0}
F_1, F_2
8
F_1
F_2
P
Z
ALU
F(P,Q)
43
Q
8
8
C_{ALUreg}
Clock
ALUreg
In
8
Out
E_{ALUreg}


Out
5
E_{PC}
Enables (E)
C_{D0}
Clock
D0
In
8
Clocks (C)
42
Out
8
R, W
E_{D0}
F_1, F_2
8
F_1
F_2
P
Z
ALU
F(P,Q)
43
Q
8
8
C_{ALUreg}
Clock
ALUreg
In
43
Out
8
E_{ALUreg}


Out
5
E_{PC}
Enables (E)
C_{D0}
Clock
D0
In
8
Clocks (C)
42
Out
8
R, W
E_{D0}
F_1, F_2
8
F_1
F_2
P
Z
ALU
F(P,Q)
8
Q
8
C_{ALUreg}
Clock
ALUreg
In
43
8
Out
E_{ALUreg}


Out
5
E_{PC}
Enables (E)
C_{D0}
Clock
D0
In
43
8
Clocks (C)
Out
8
R, W
E_{D0}
F_1, F_2
F_1
F_2
P
8
Z
ALU
F(P,Q)
Q
8
8
C_{ALUreg}
Clock
ALUreg
In
43
Out
8
E_{ALUreg}

H4: how do D0 and ALU interact when control signals for example 0 are asserted?

Work in pairs.

One person is the CU.

The other is the components.

Work through H4 and execute the program, as if you were "toggling switches" to do the job of the CU.

Do you get the right answer at the end?

When you finish, swap roles & do it again.

Questions?

Please bring the handouts back next time!