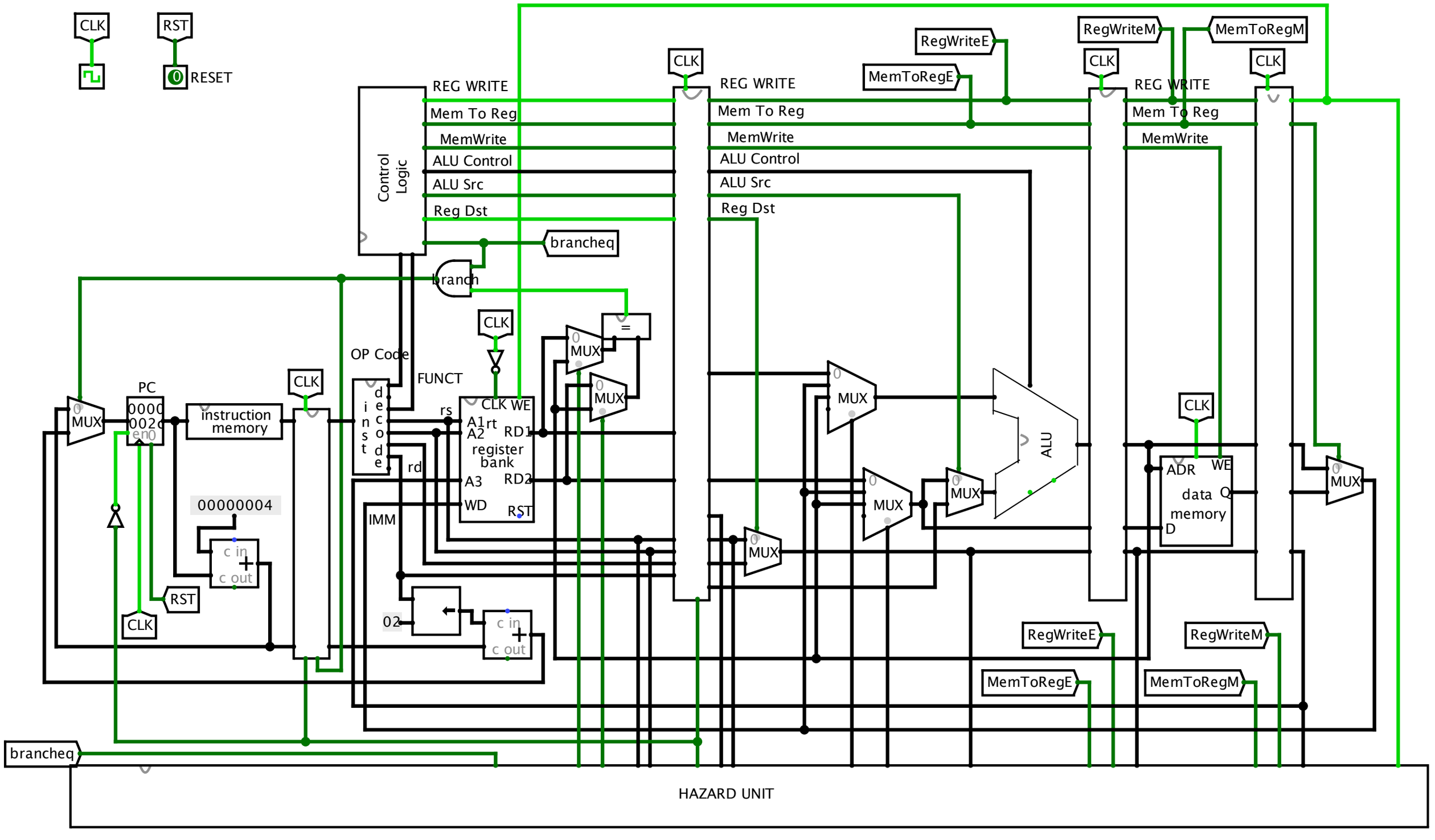
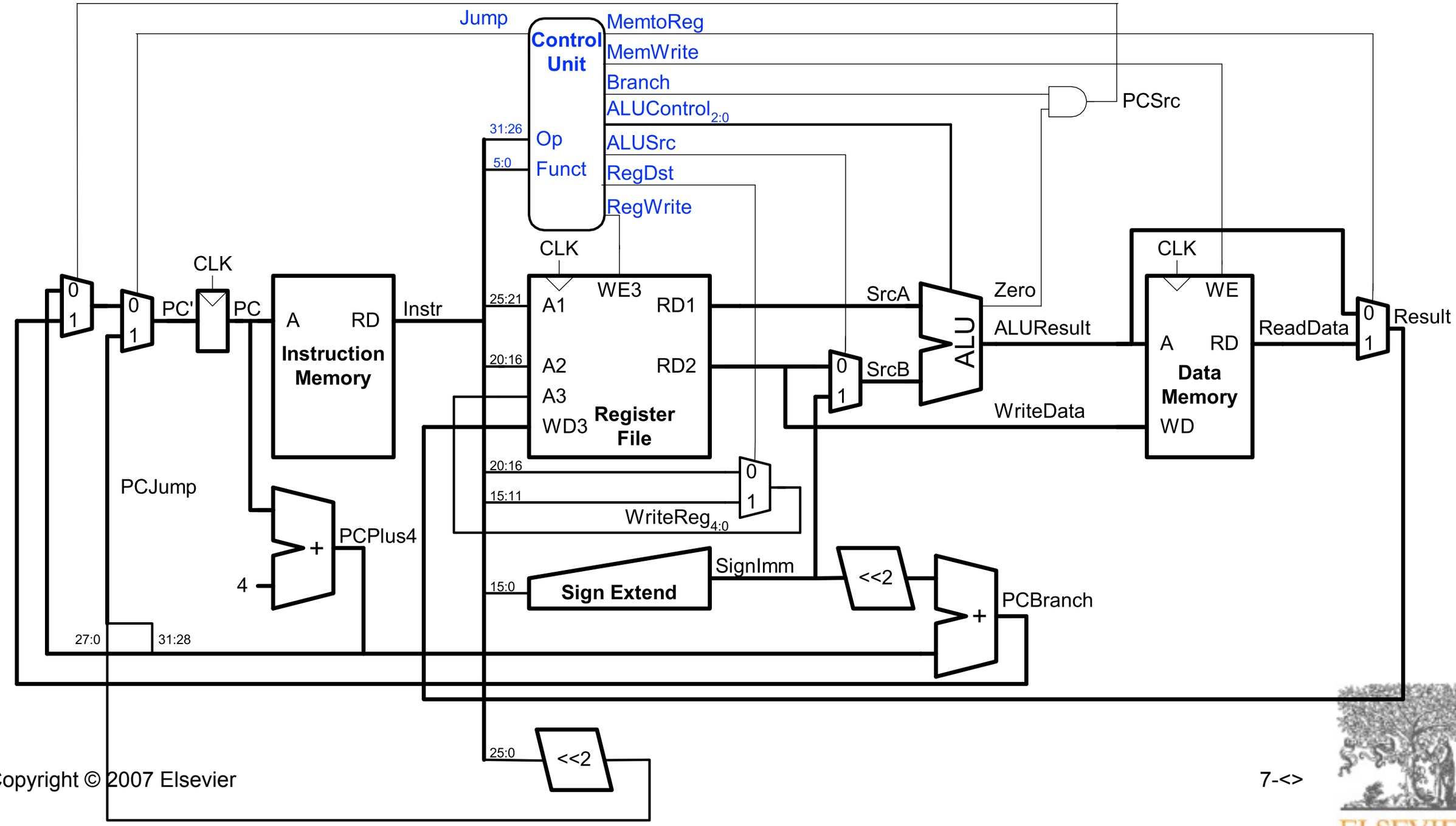
Announcements

Control circuit complete by Tuesday

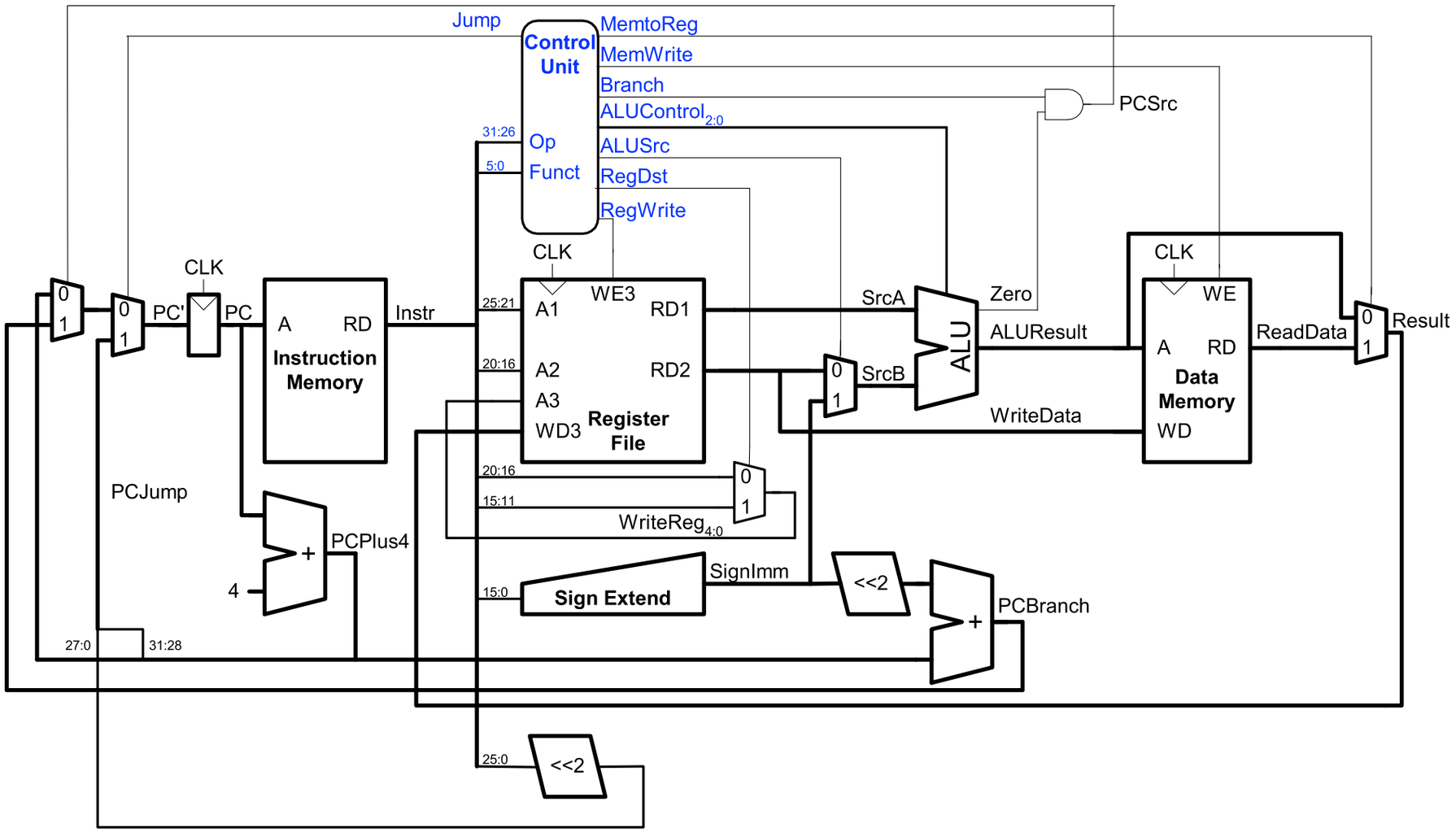
Pipelining


CLK
RST
RESET
REG WRITE
Mem To Reg
MemWrite
ALU Control
ALU Src
Reg Dst
Control Logic
brancheq
branch
CLK
REG WRITE
Mem To Reg
MemWrite
ALU Control
ALU Src
Reg Dst
RegWriteE
MemToRegE
RegWriteM
MemToRegM
CLK
REG WRITE
Mem To Reg
MemWrite
CLK
OP Code
FUNCT
PC
0000 0020
en0
instruction memory
inst
decode
rs
rt
rd
IMM
CLK WE
A1
A2
register bank
RD1
RD2
A3
WD
RST
00000004
c in
c out
02
MUX
=
ALU
ADR
WE
data memory
Q
D
RegWriteE
MemToRegE
RegWriteM
MemToRegM
HAZARD UNIT

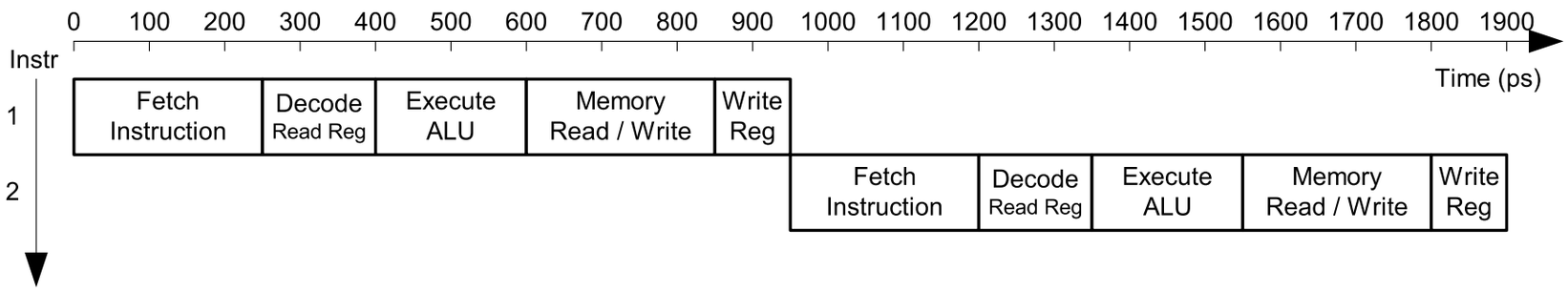
Review: Single-Cycle MIPS Processor



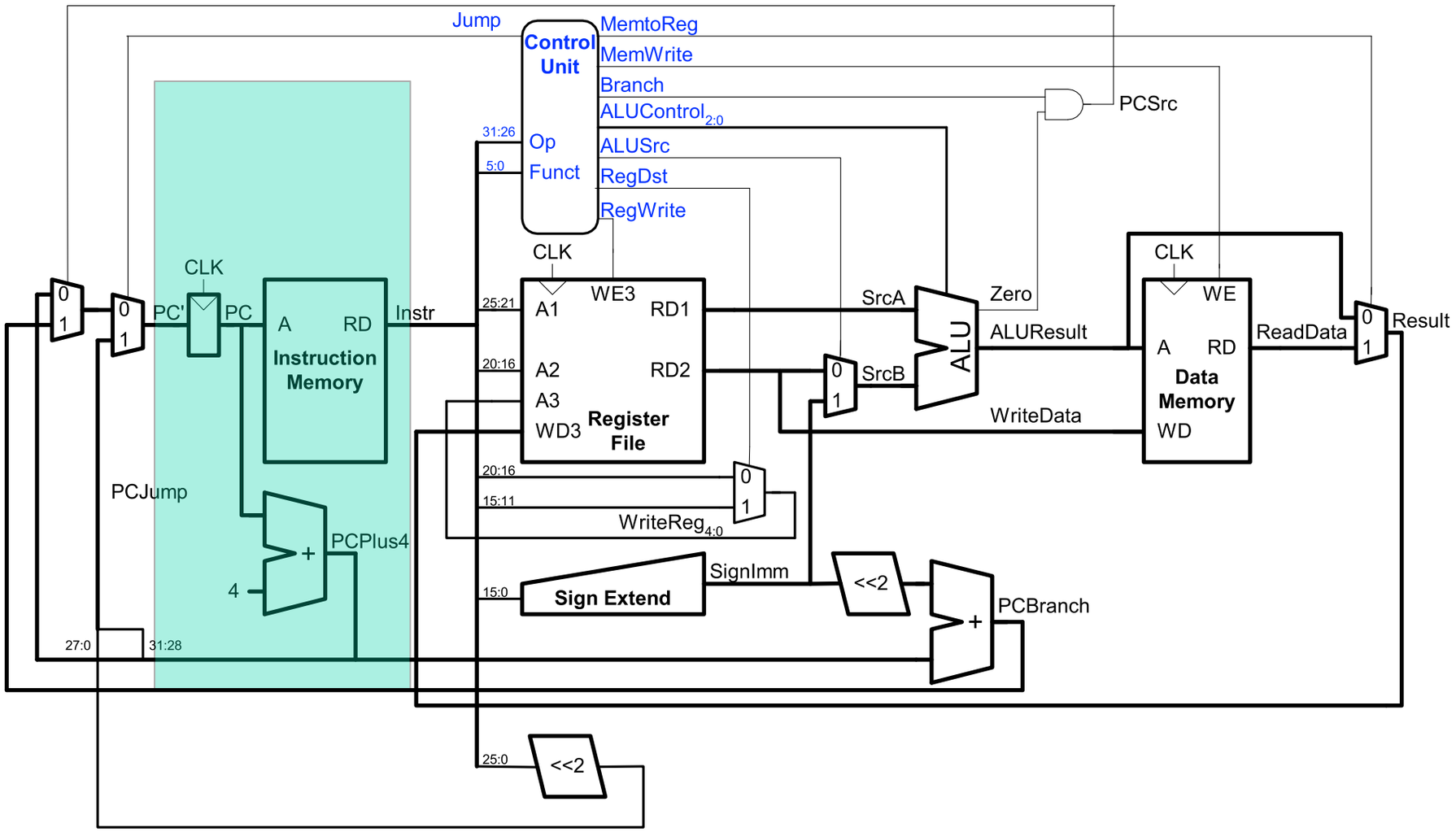
Review: Single-Cycle MIPS Processor



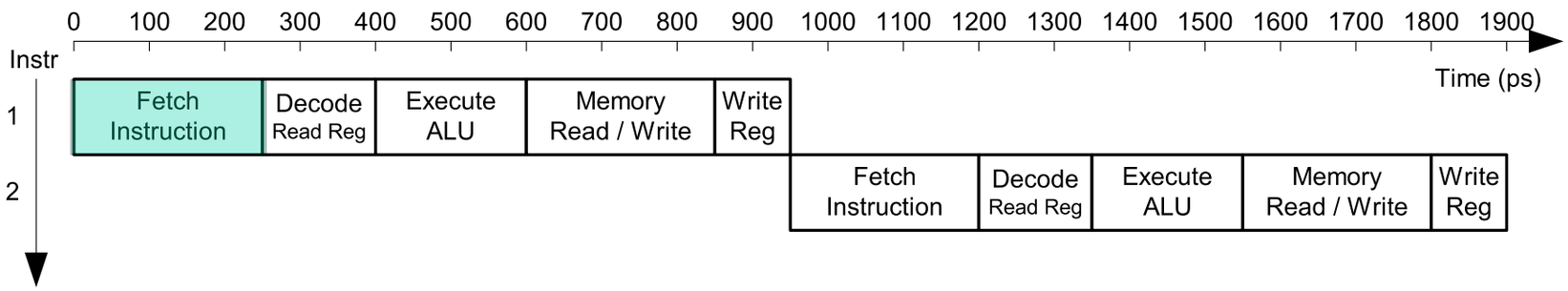
Single-Cycle



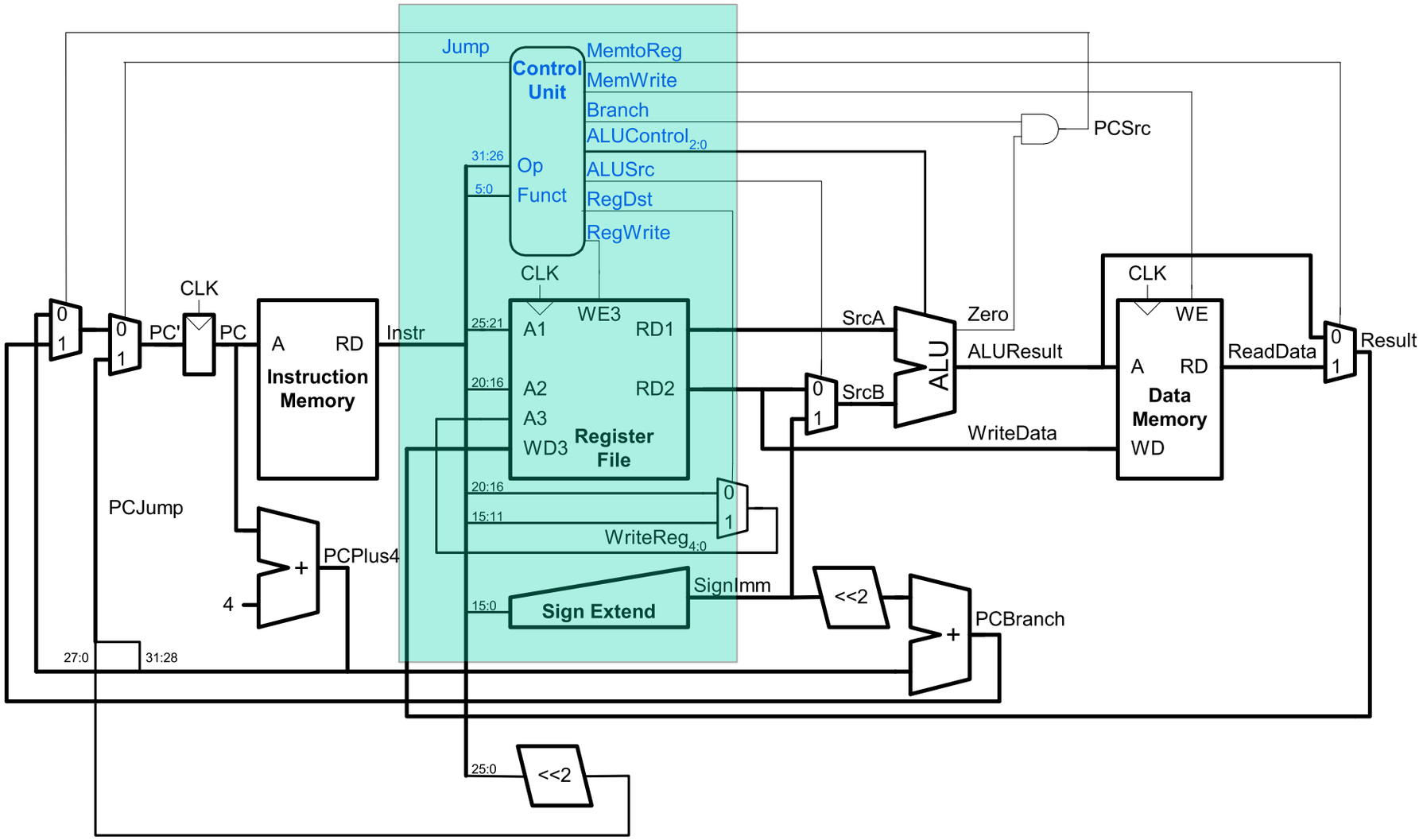
Review: Single-Cycle MIPS Processor



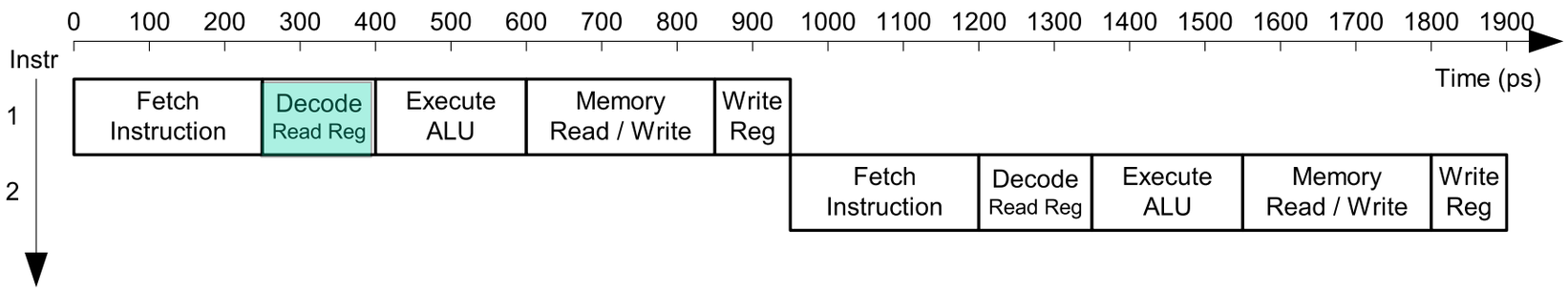
Single-Cycle



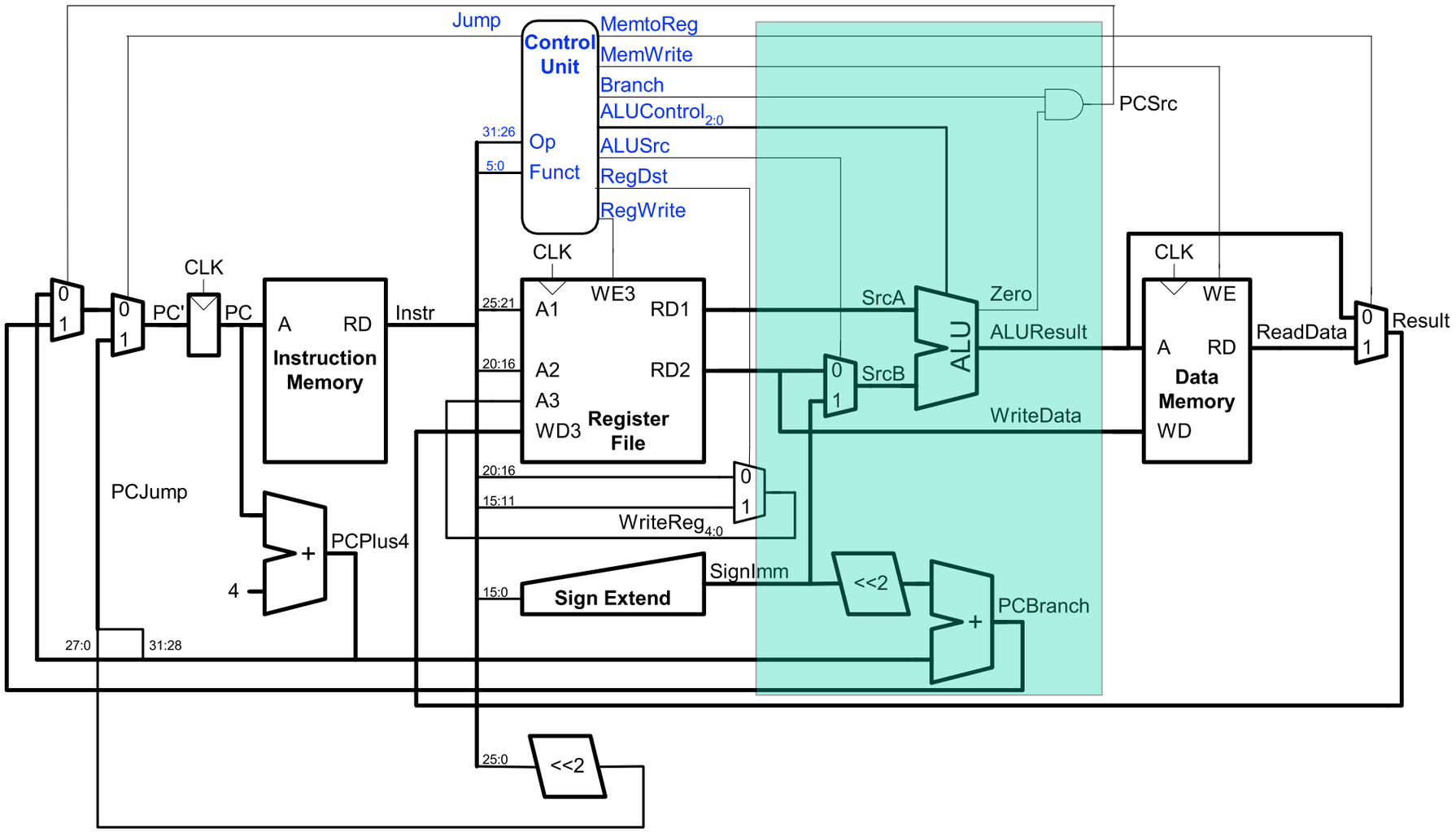
Review: Single-Cycle MIPS Processor



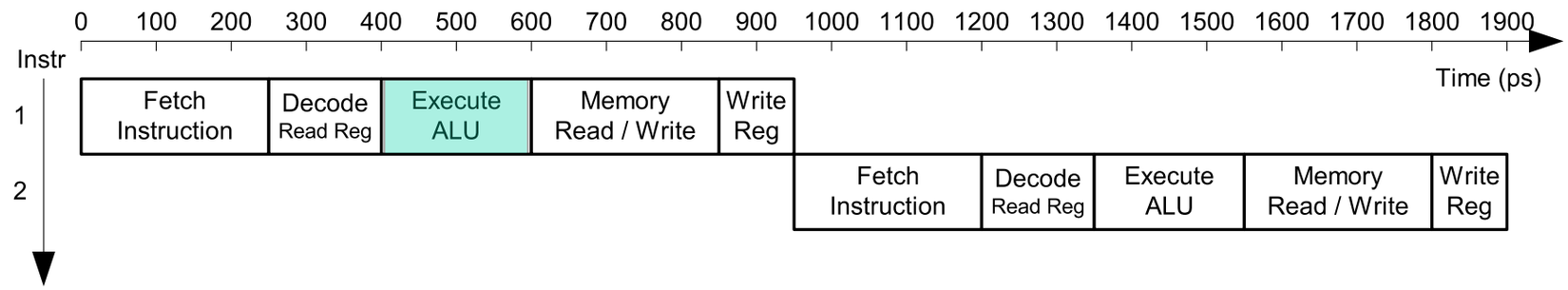
Single-Cycle



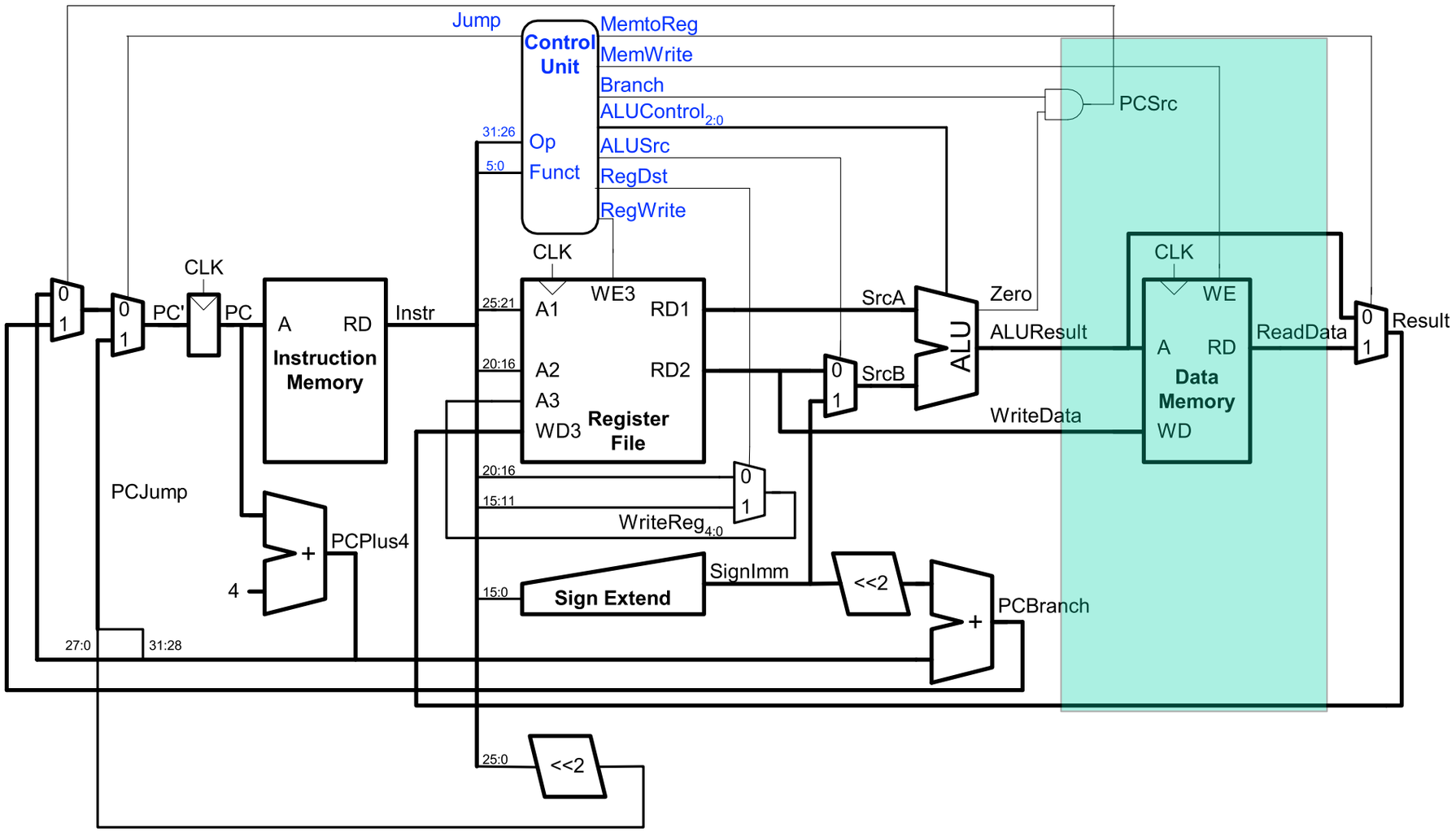
Review: Single-Cycle MIPS Processor



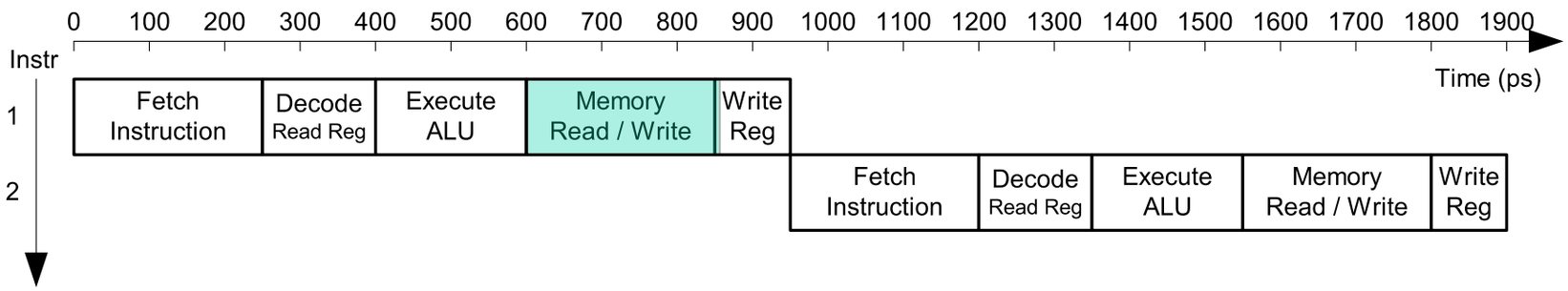
Single-Cycle



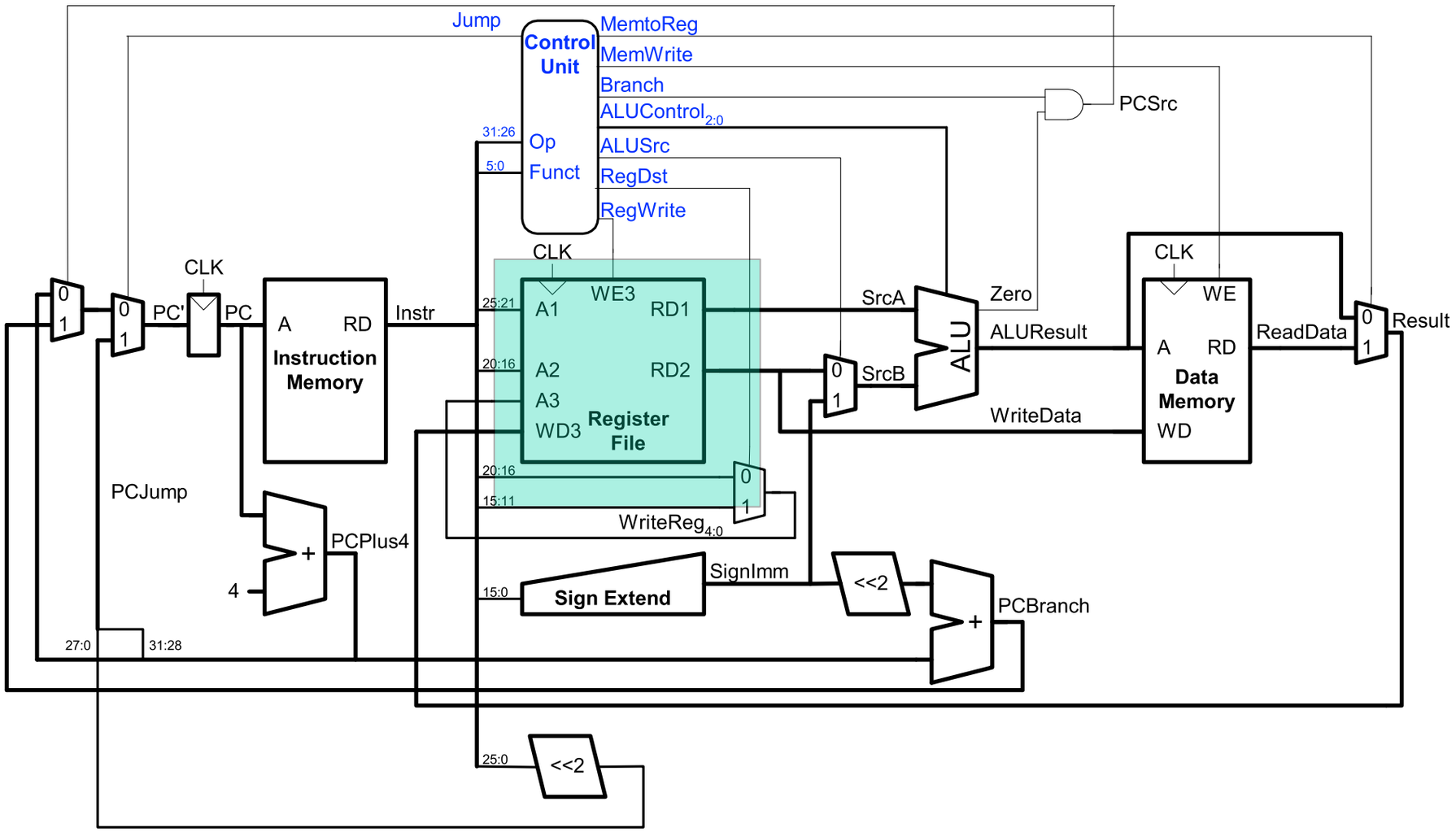
Review: Single-Cycle MIPS Processor



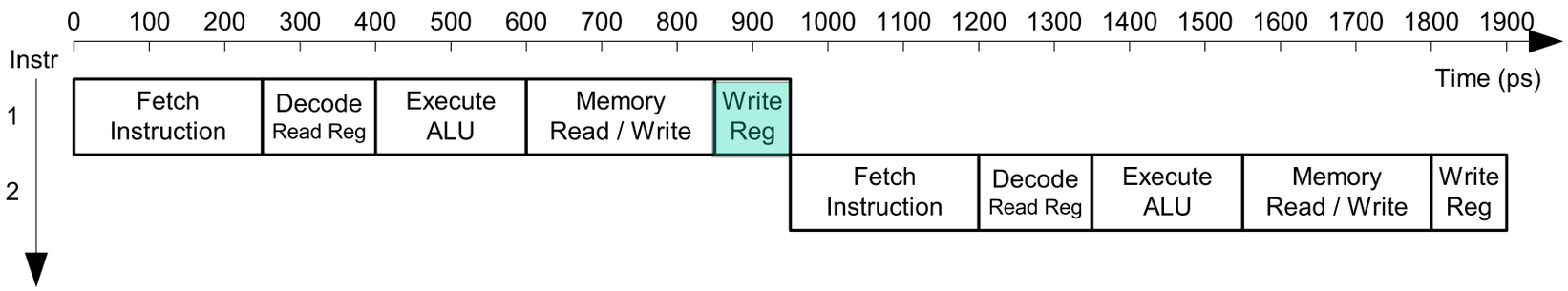
Single-Cycle



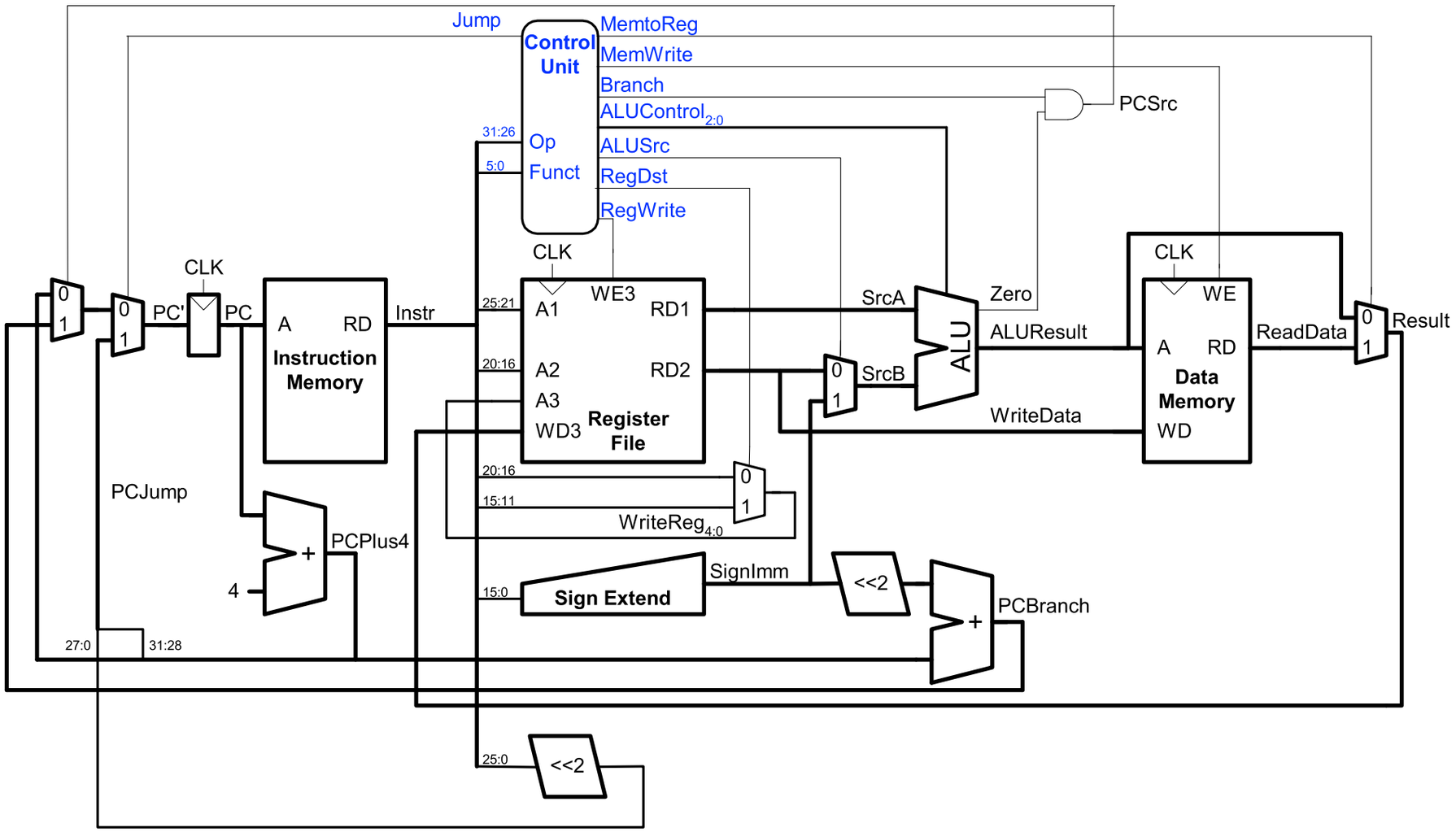
Review: Single-Cycle MIPS Processor

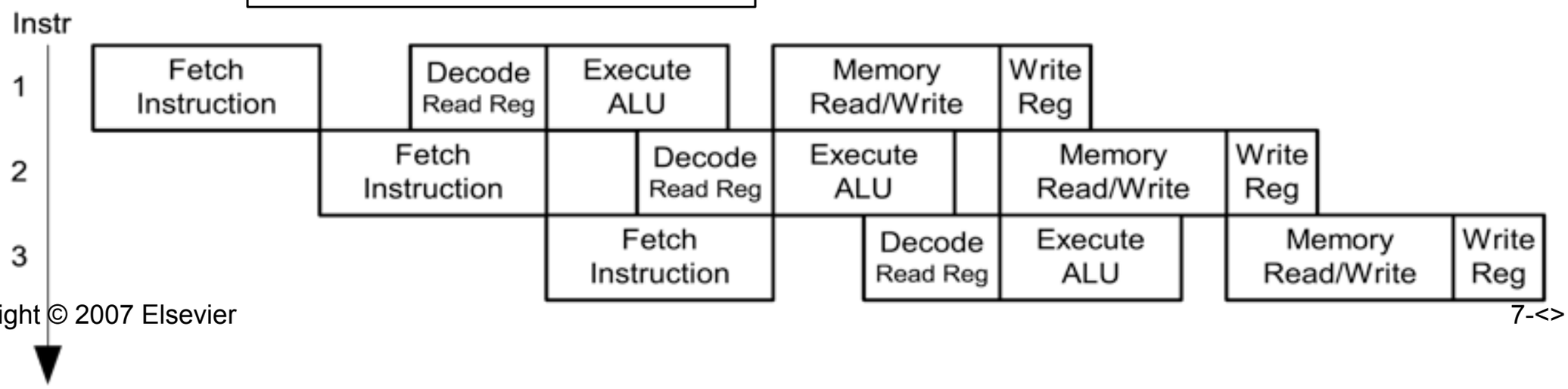


Single-Cycle

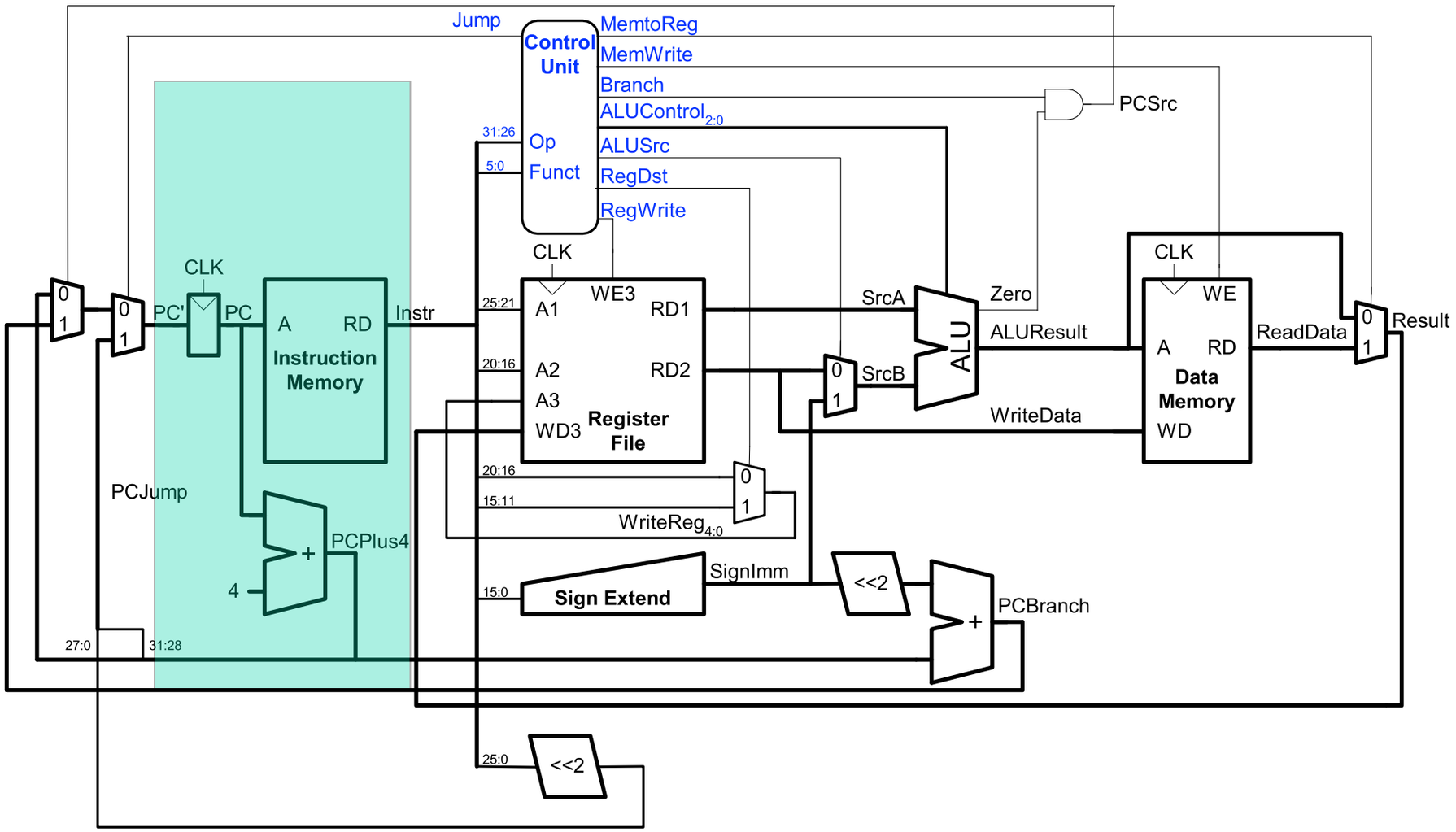


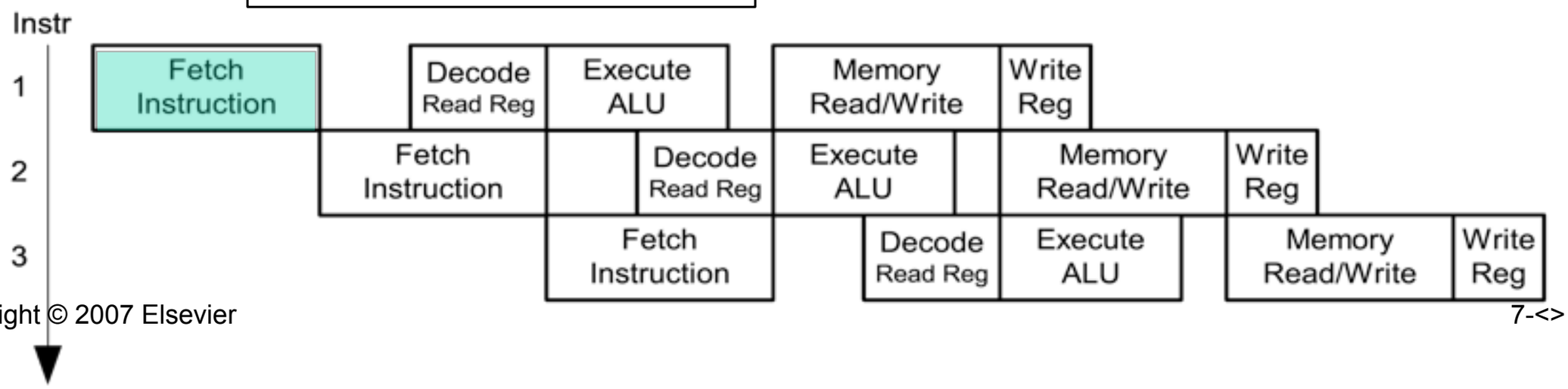
Review: Single-Cycle MIPS Processor



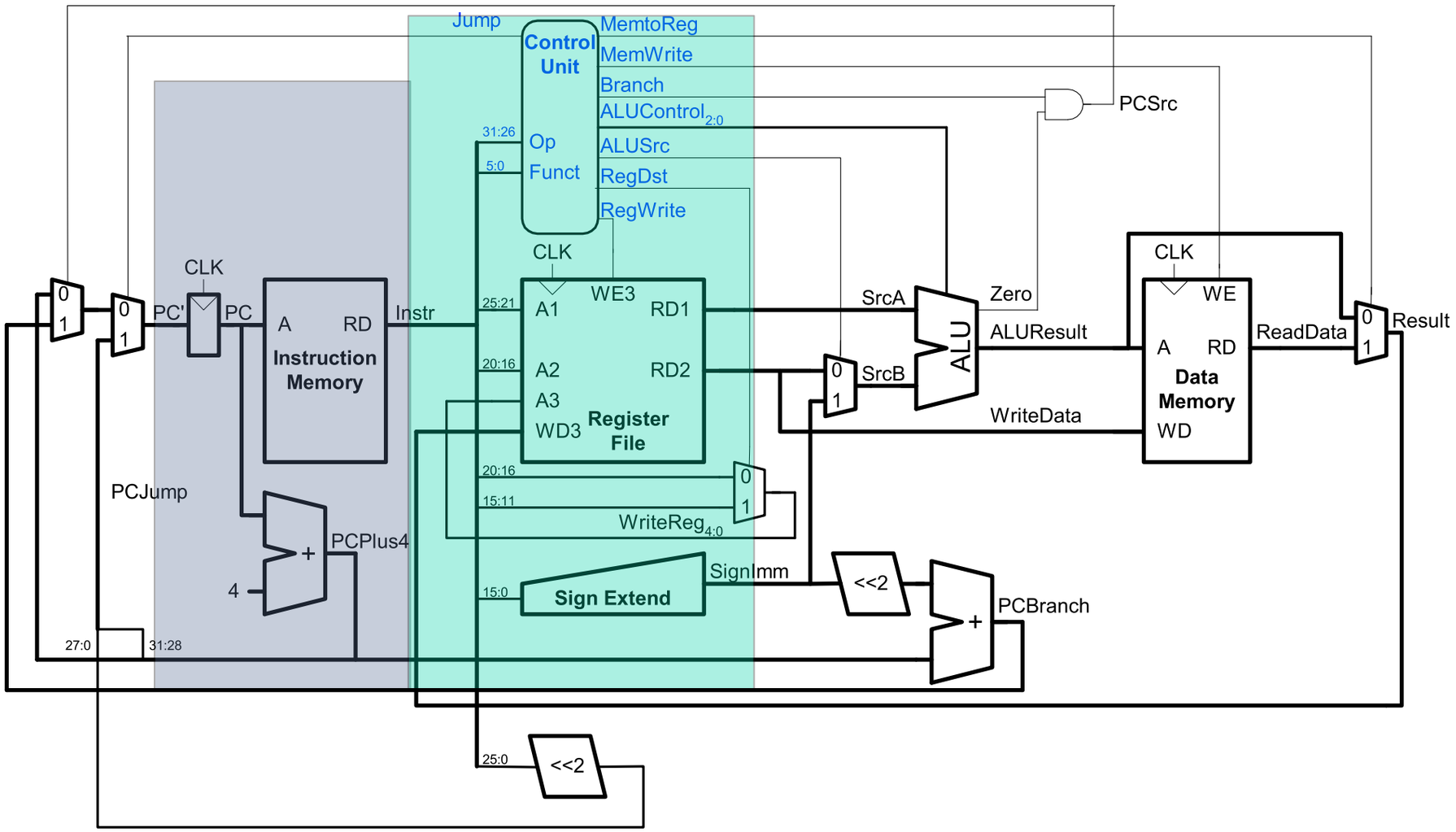


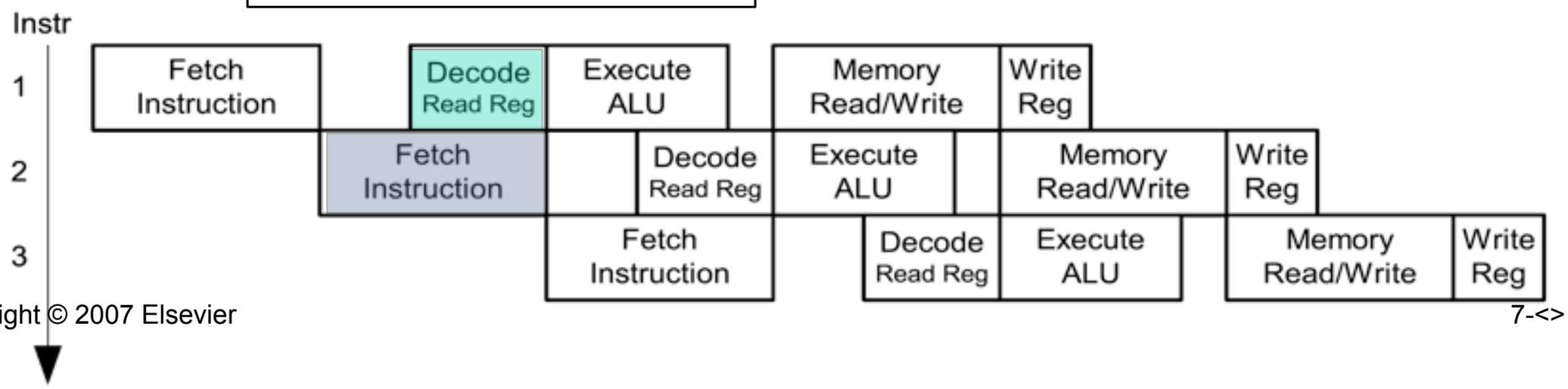
Review: Single-Cycle MIPS Processor



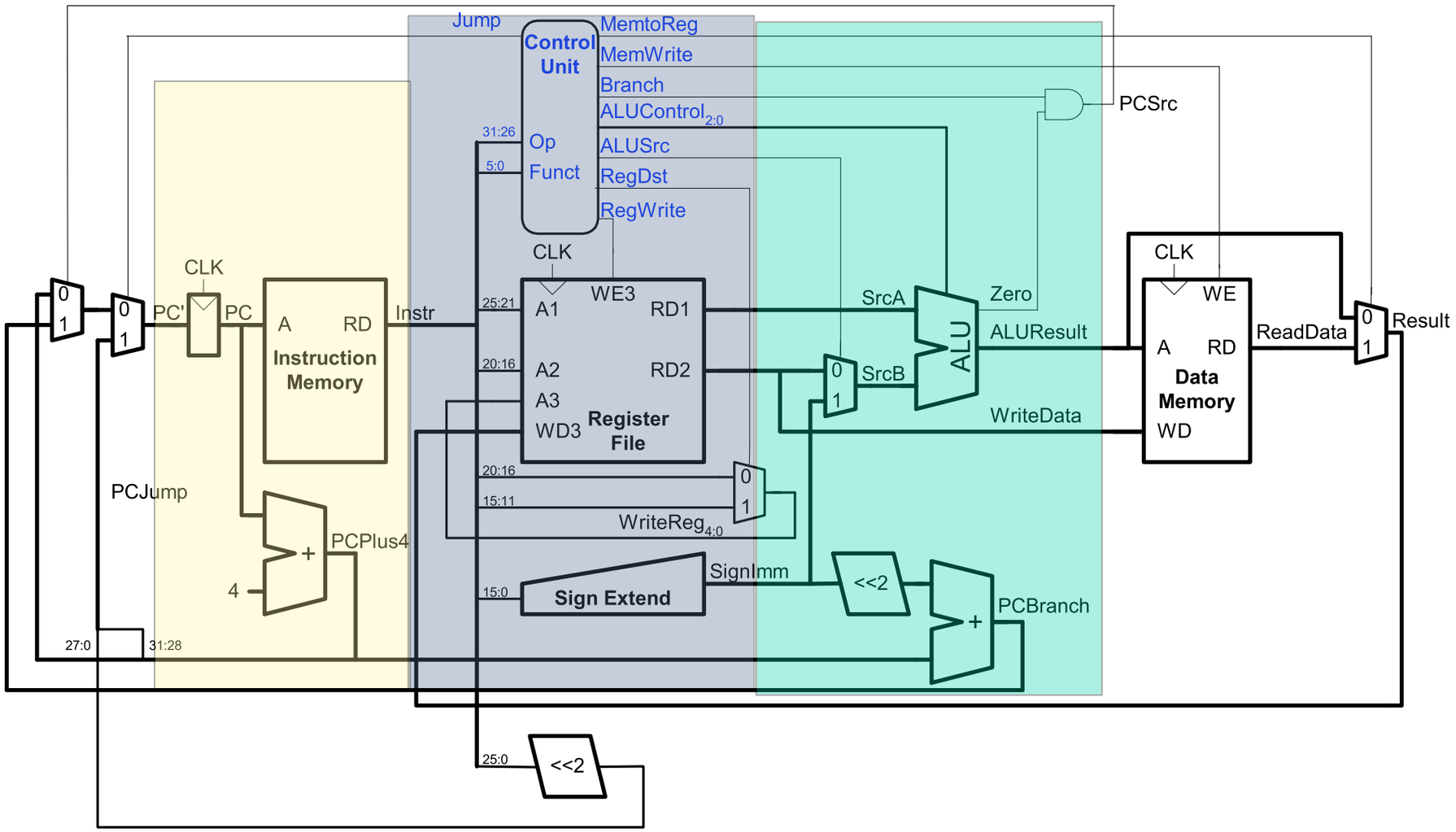


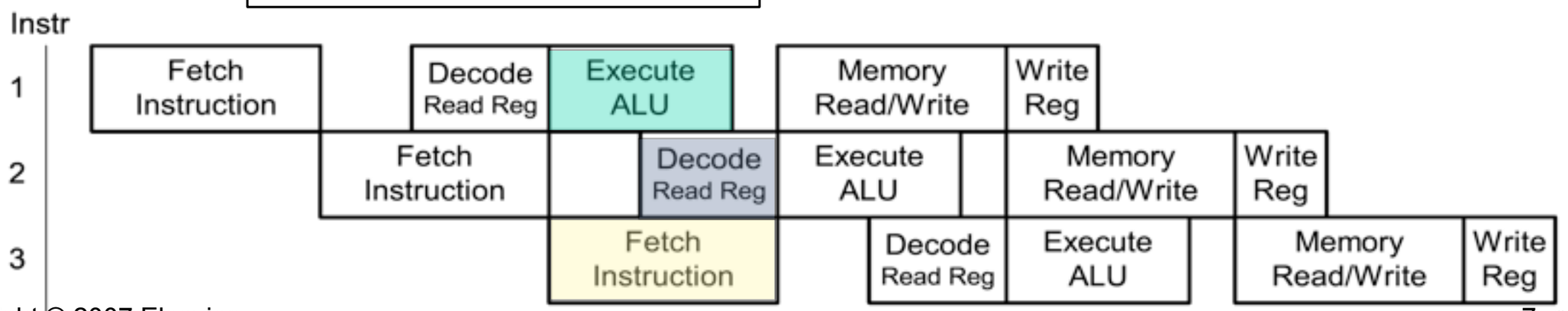
Review: Single-Cycle MIPS Processor





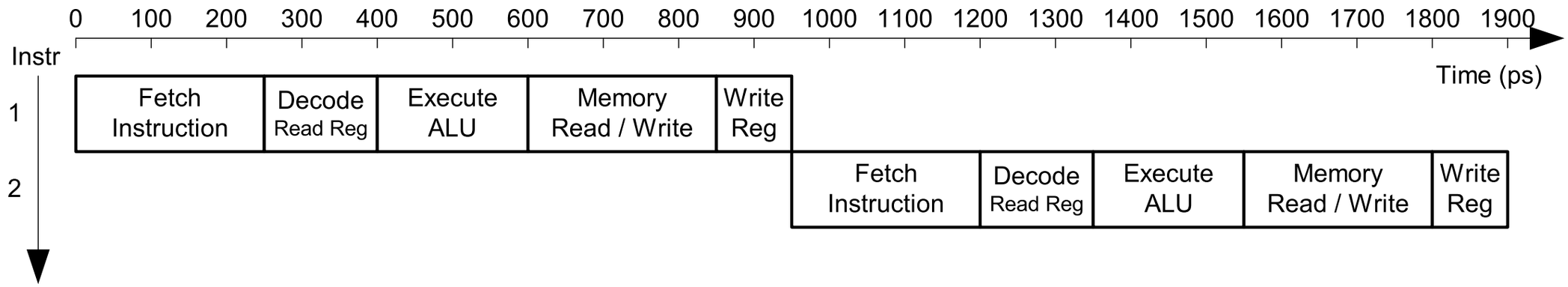
Review: Single-Cycle MIPS Processor



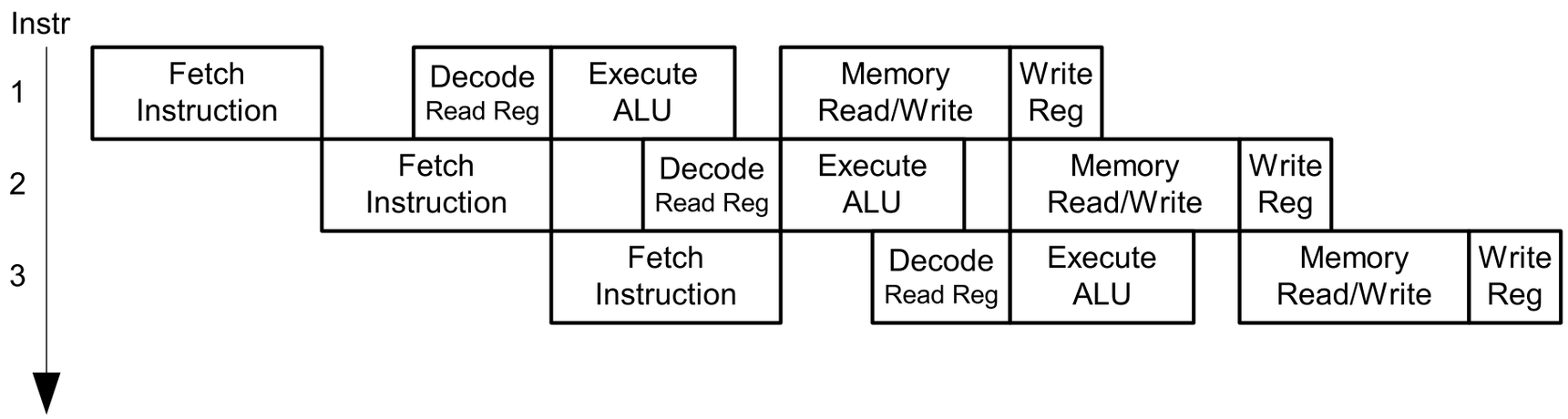


Single-Cycle vs. Pipelined Performance

Single-Cycle

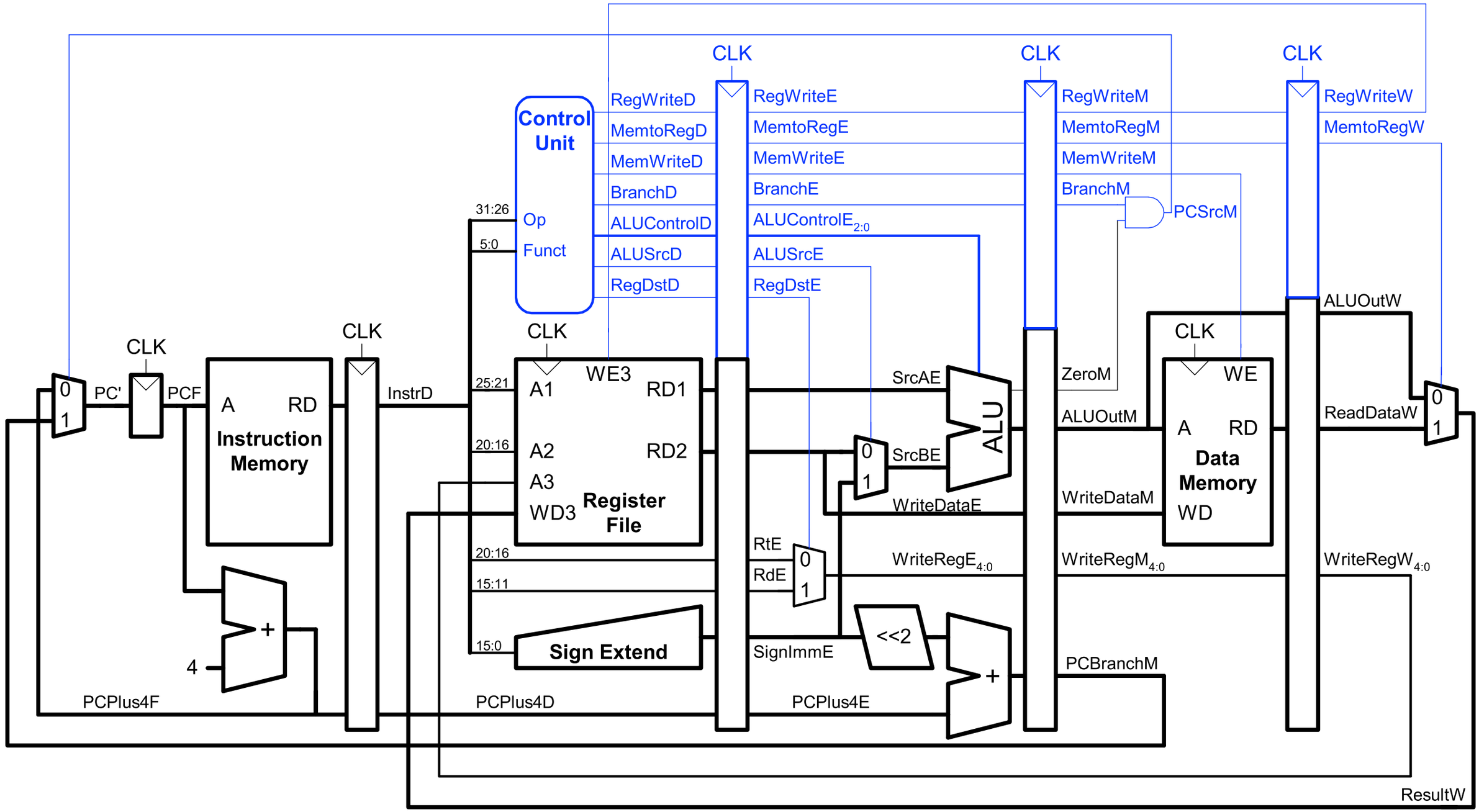


Pipelined



ELSEVIER

Pipelined Processor

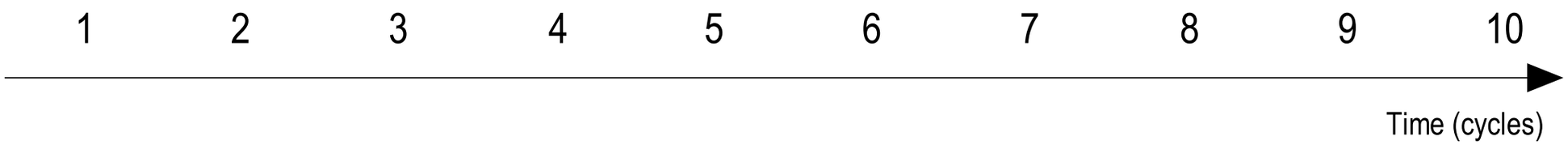


Same control unit as single-cycle processor

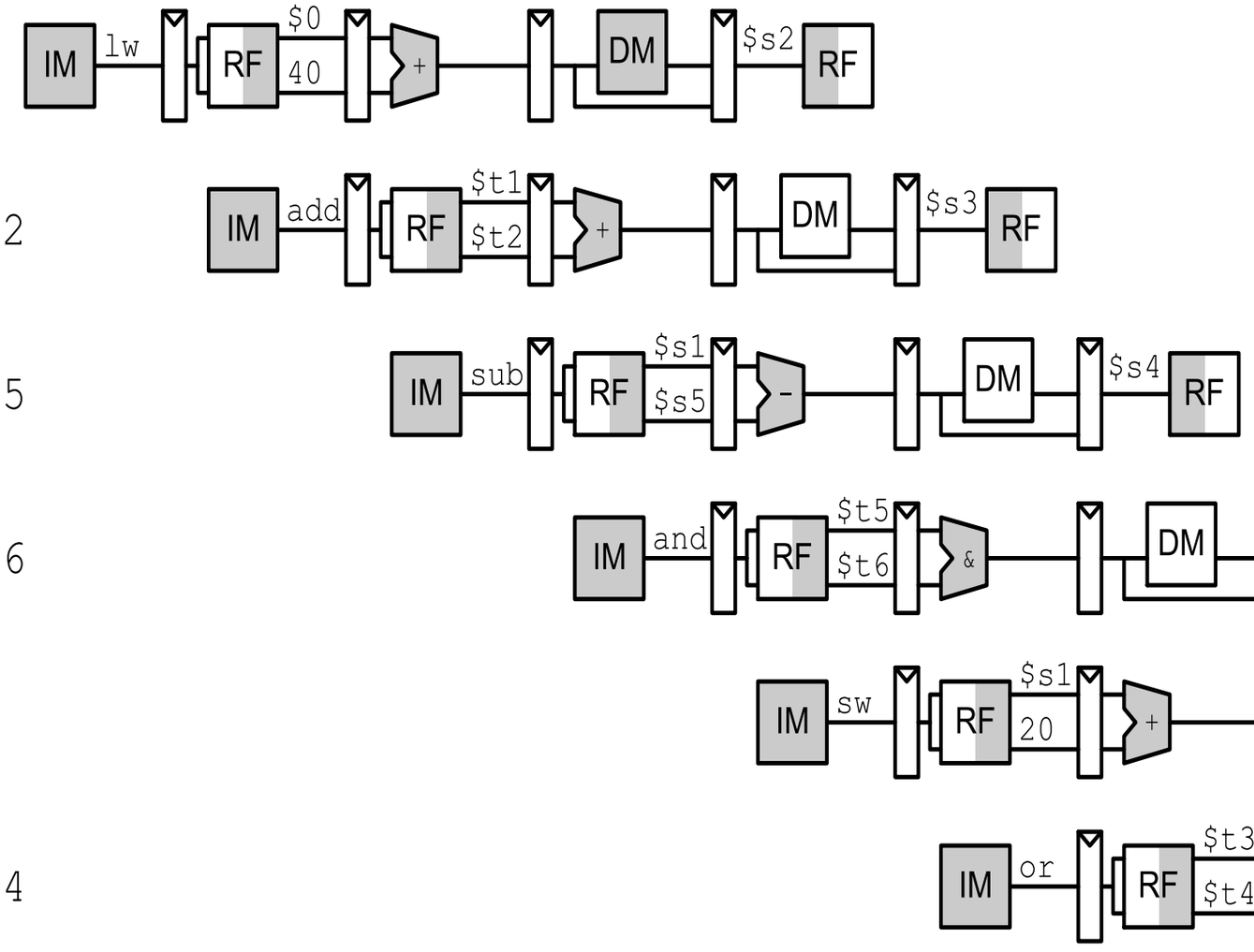
Signal preserved for pipeline stage where used

ELSEVIER

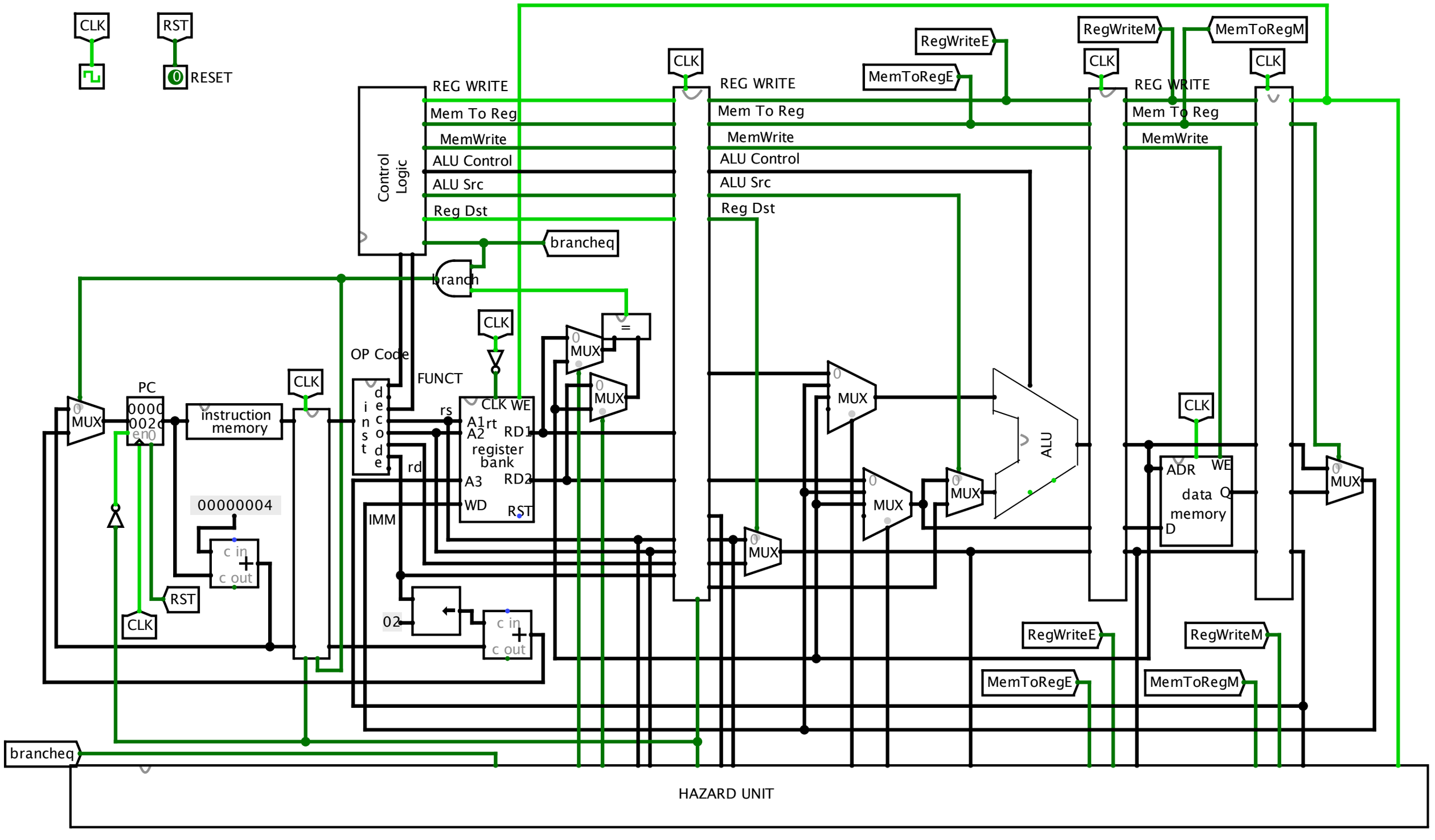
Pipelining Abstraction




```
lw  $s2, 40($0)
add $s3, $t1, $t2
sub $s4, $s1, $s5
and $s5, $t5, $t6
sw  $s6, 20($s1)
or  $s7, $t3, $t4
```



Pipelining


CLK
RST
RESET
REG WRITE
Mem To Reg
MemWrite
ALU Control
ALU Src
Reg Dst
Control Logic
brancheq
branch
CLK
REG WRITE
Mem To Reg
MemWrite
ALU Control
ALU Src
Reg Dst
RegWriteE
MemToRegE
RegWriteM
MemToRegM
CLK
REG WRITE
Mem To Reg
MemWrite
CLK
OP Code
FUNCT
PC
0000 0020
en0
instruction memory
inst
decode
rs
rt
rd
IMM
CLK WE
A1
A2
register bank
RD1
RD2
A3
WD
RST
00000004
c in
c out
02
MUX
=
ALU
ADR
WE
data memory
Q
D
RegWriteE
MemToRegE
RegWriteM
MemToRegM
HAZARD UNIT

STAGE F

STAGE D

STAGE E

STAGE M

STAGE W

CLK

RST

RESET

REG WRITE

Mem To Reg

MemWrite

ALU Control

ALU Src

Reg Dst

Control Logic

branch

brancheq

OP Code

FUNCT

PC

0000
0004

MUX

en0

instruction memory

00000004

c in

c out

RST

CLK

decode

inst

rs

rt

rd

IMM

02

CLK WE

A1

A2

register bank

A3

WD

RD1

RD2

RST

=

MUX

RegWriteE

MemToRegE

ALU

RegWriteM

MemToRegM

ADR

WE

data memory

D

Q

HAZARD UNIT

add $s3, $t1, $t2

lw $s2,40($0)

STAGE F

STAGE D

STAGE E

STAGE M

STAGE W

CLK

RST

RESET

REG WRITE

Mem To Reg

MemWrite

ALU Control

ALU Src

Reg Dst

Control Logic

brancheq

branch

OP Code

FUNCT

PC

MUX

0000 0008

en0

instruction memory

00000004

c in

c out

RST

CLK

inst decode

rs

rt

rd

IMM

02

CLK WE

A1

A2

register bank

A3

WD

RD1

RD2

RST

=

MUX

RegWriteE

MemToRegE

RegWriteM

MemToRegM

ALU

ADR

WE

data memory

D

Q

HAZARD UNIT

sub $s4, $s1, $s5

add $s3, $t1, $t2

lw $s2,40($0)

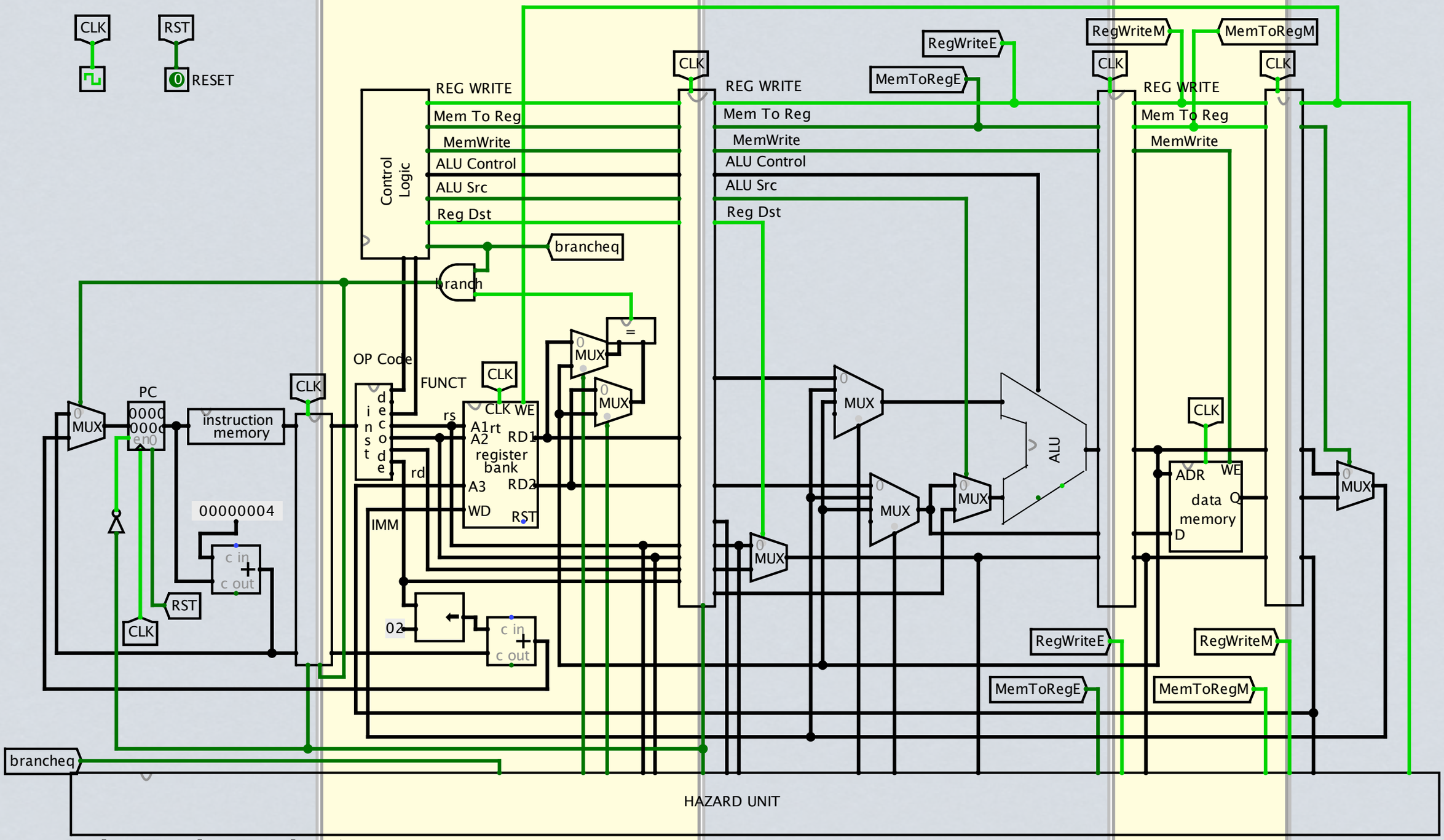
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



and $s5, $t5, $t6

sub $s4, $s1, $s5

add $s3, $t1, $t2

lw $s2,40($0)

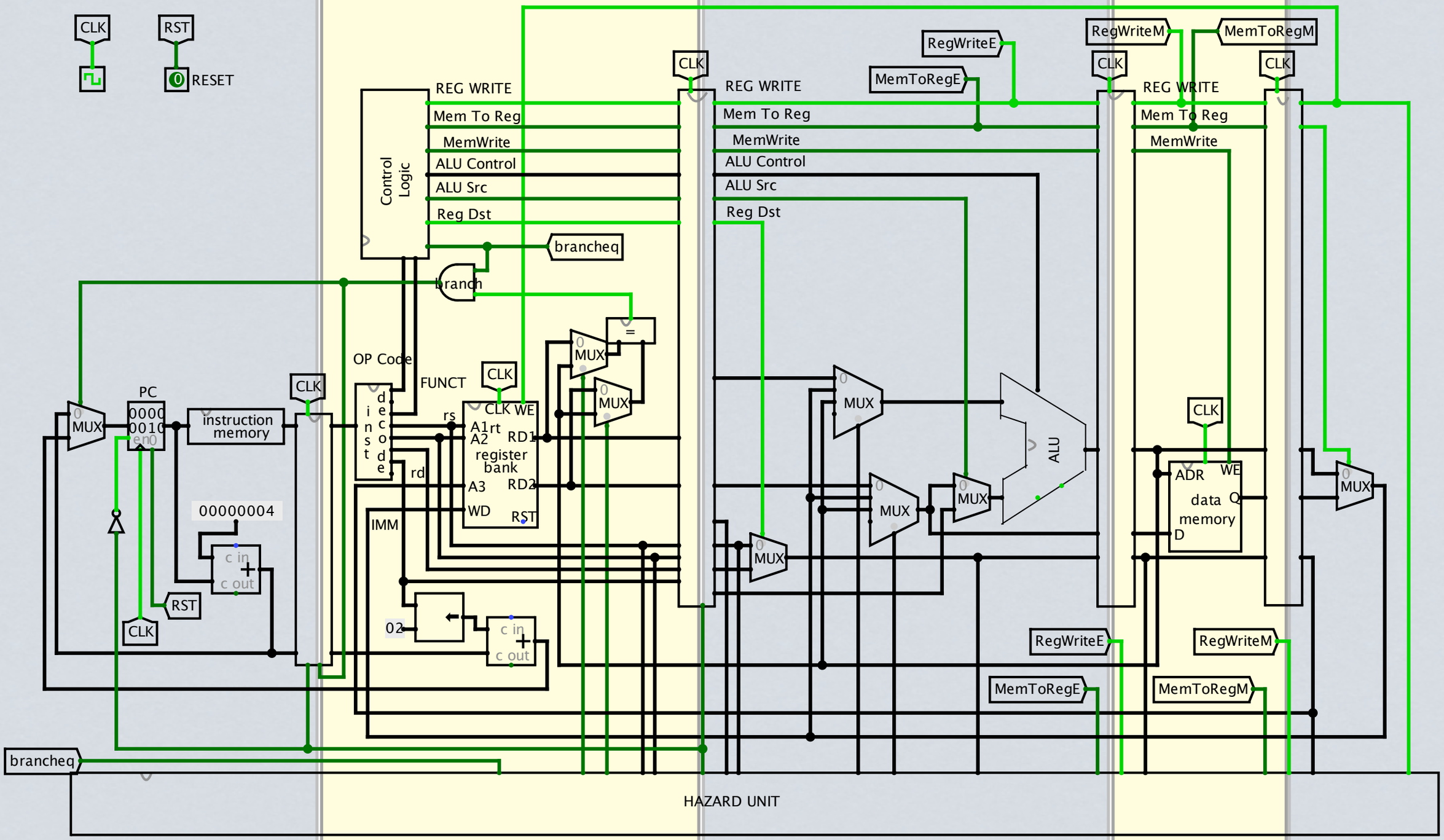
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



sw $s6, 20($s1)

and $s5, $t5, $t6

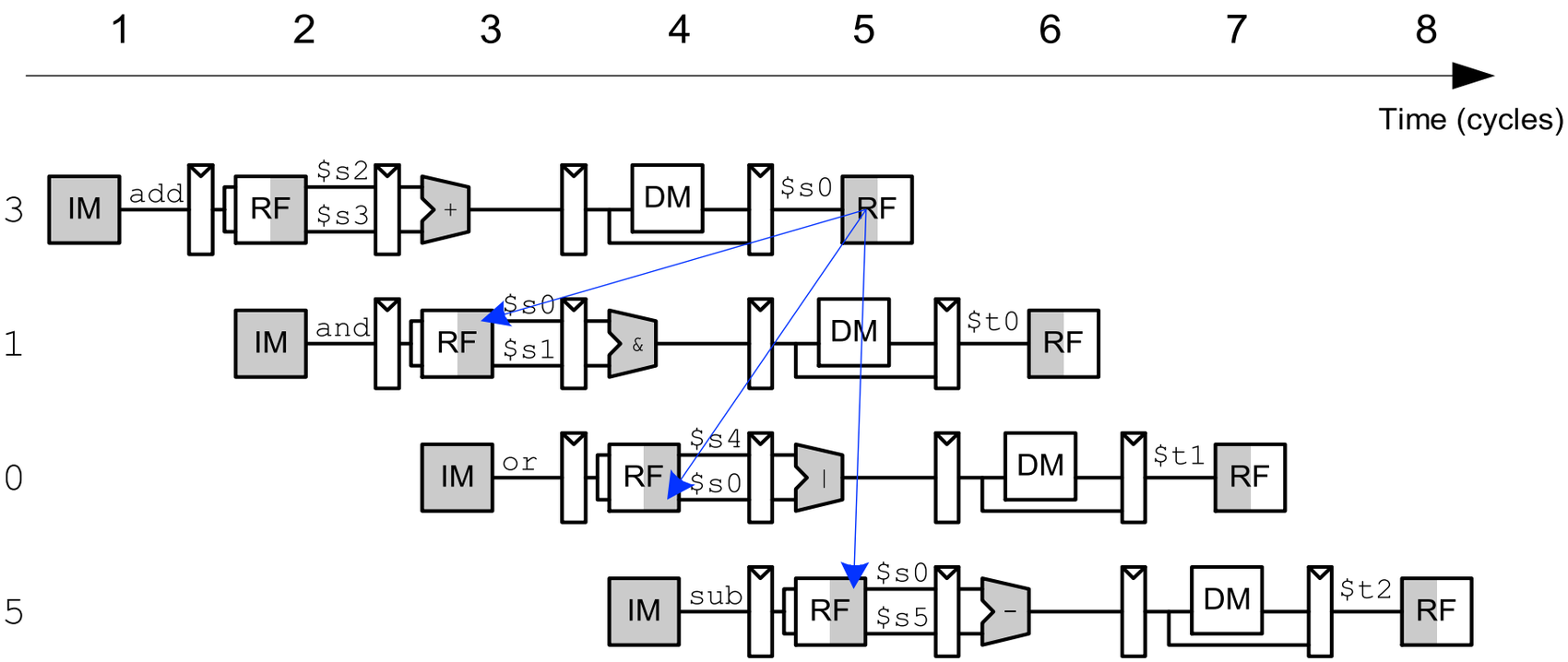
sub $s4, $s1, $s5

add $s3, $t1, $t2

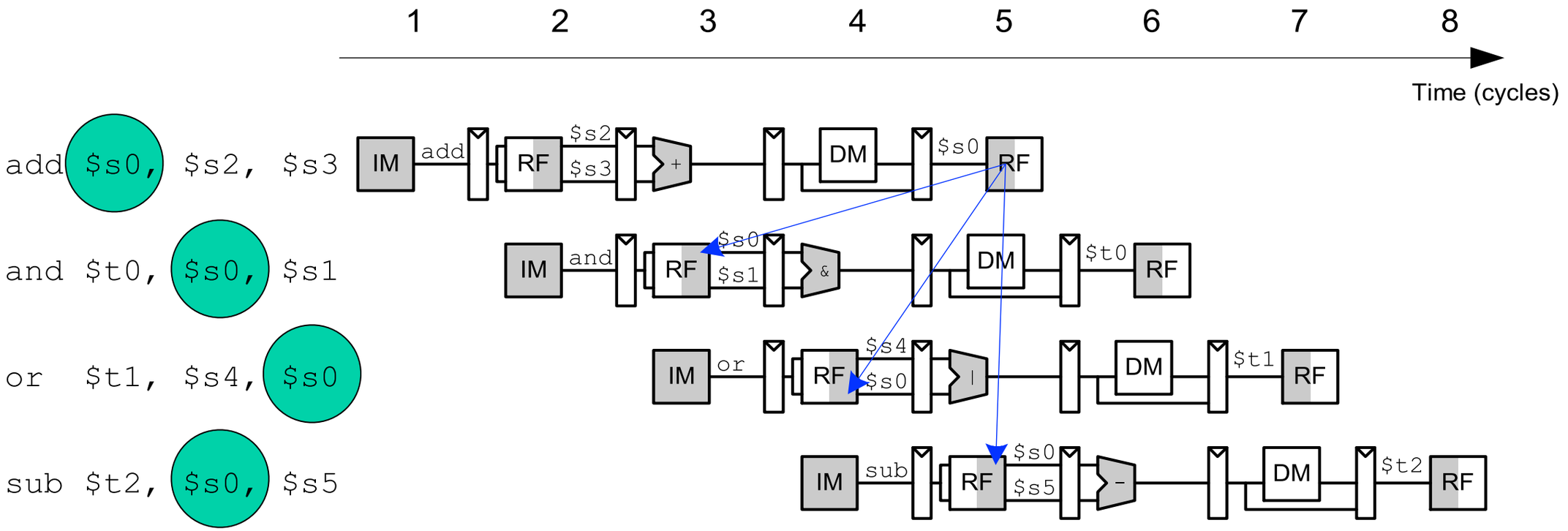
lw $s2,40($0)

Data Hazards and Forwarding

```
add $s0, $s2, $s3

and $t0, $s0, $s1

or  $t1, $s4, $s0

sub $t2, $s0, $s5
```



Data Hazards and Forwarding



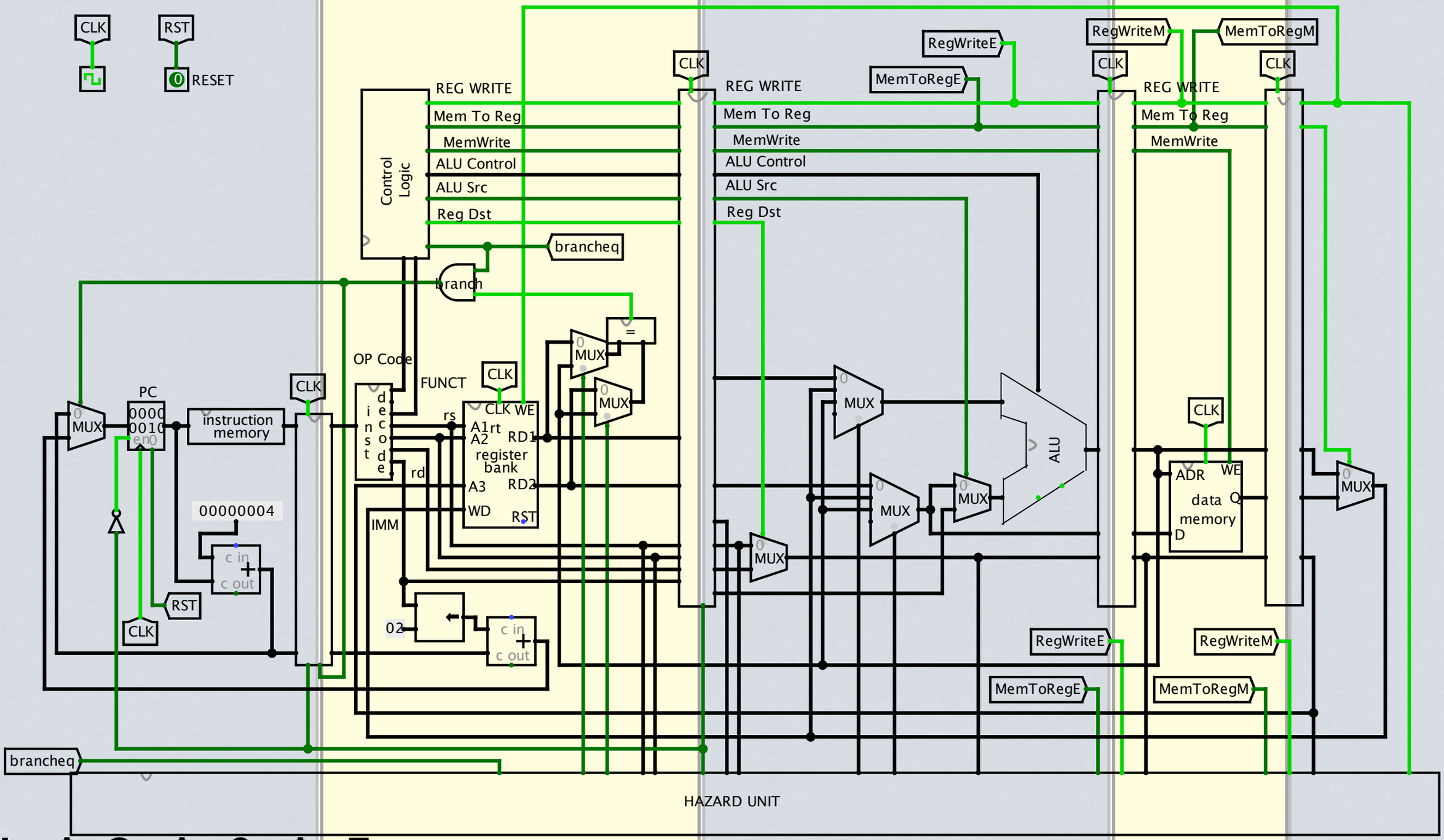
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W

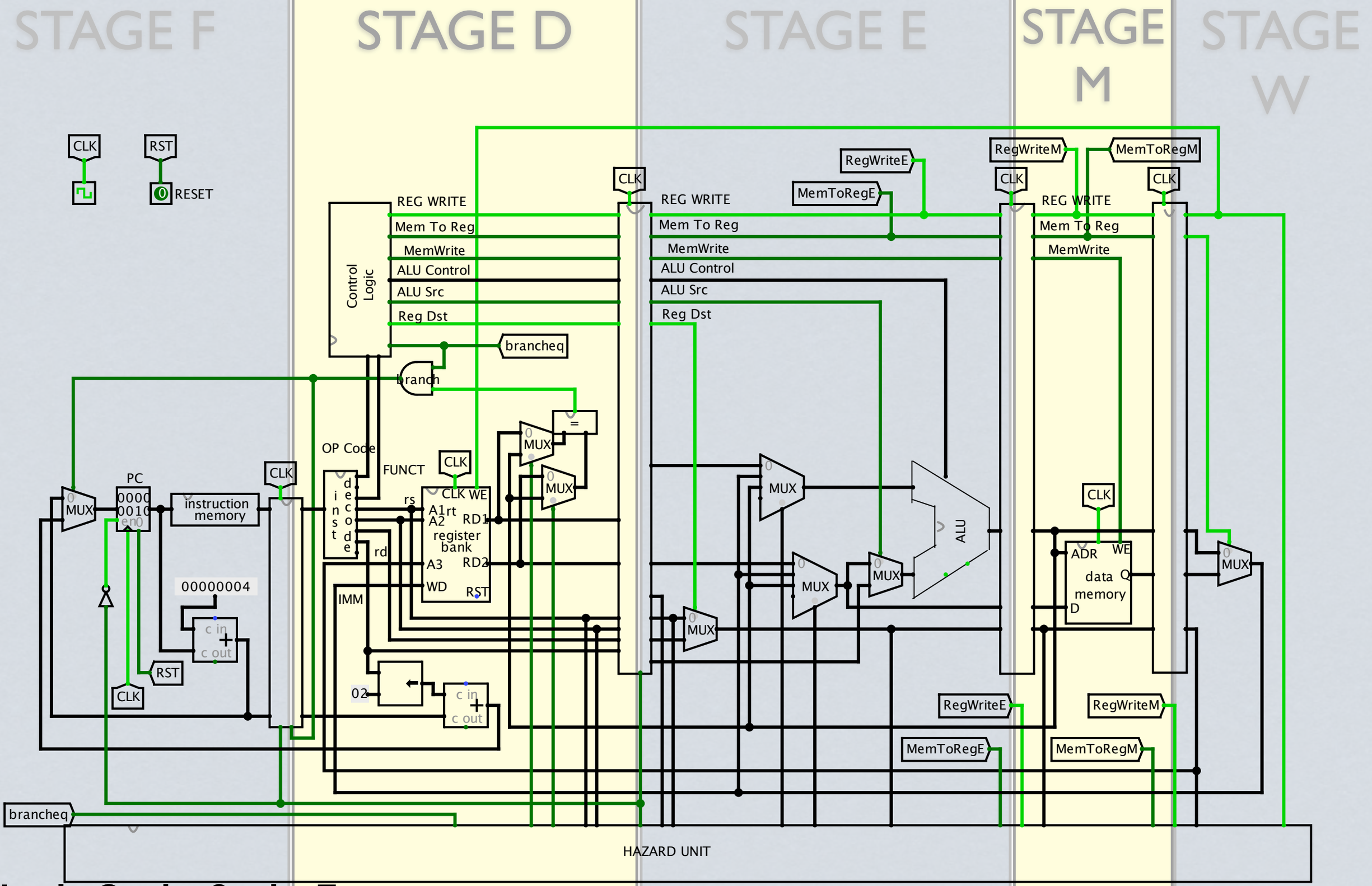


sub $t2, $s0, $s5

or $t1,$s4,$s0

and $t0, $s0, $s1

add $s0, $s2, $s3



sub $t2, $s0, $s5

or $t1,$s4,$s0

and $t0, $s0, $s1

add $s0, $s2, $s3

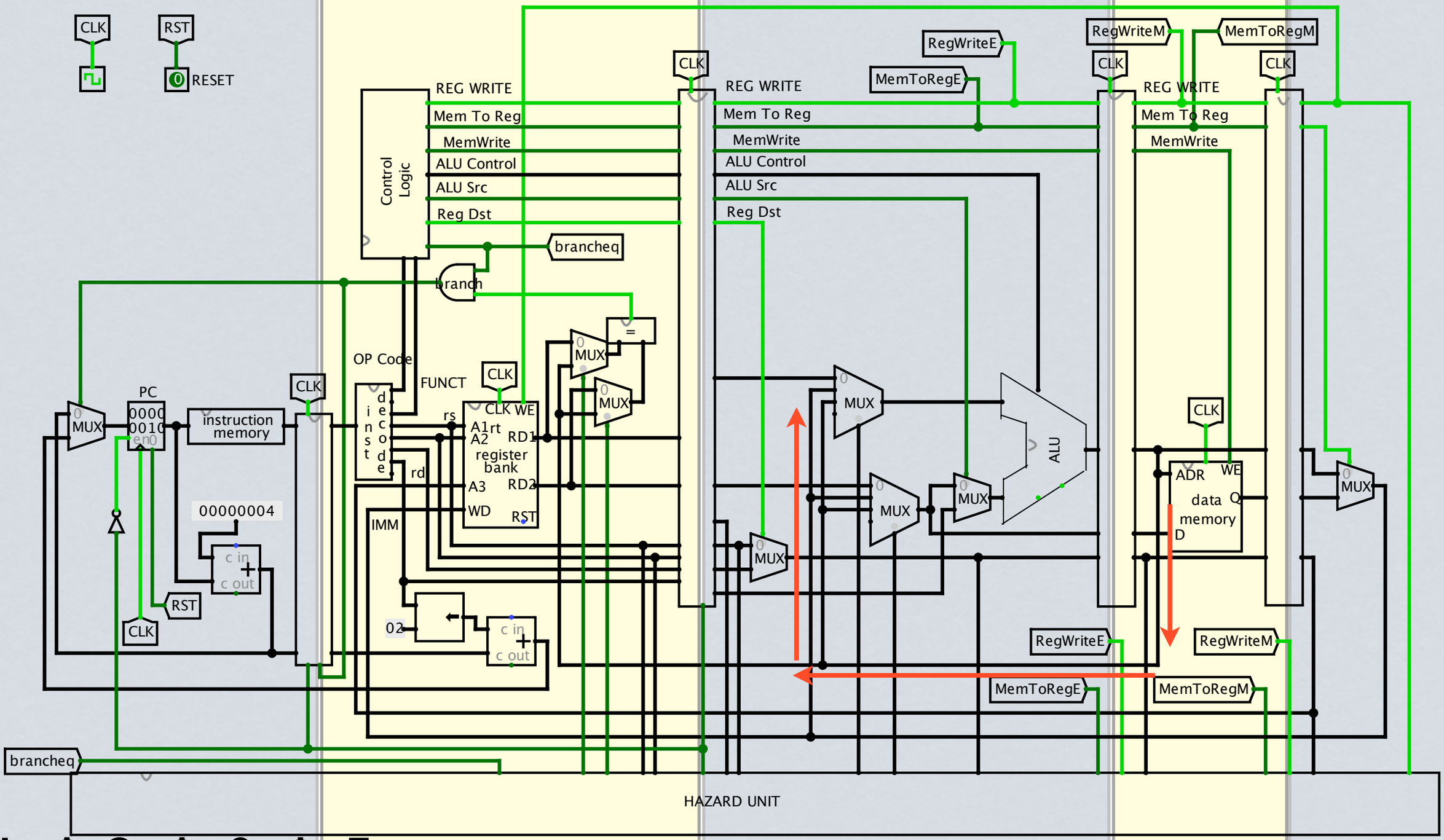
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



sub $t2, $s0, $s5

or $t1,$s4,$s0

and $t0, $s0, $s1

add $s0, $s2, $s3

STAGE F

STAGE D

STAGE E

STAGE M

STAGE W

CLK

RST

RESET

REG WRITE

Mem To Reg

MemWrite

ALU Control

ALU Src

Reg Dst

Control Logic

brancheq

branch

PC

MUX

0000
0010

en0

instruction memory

00000004

c in
c out

RST

CLK

OP Code

FUNCT

inst decode

rs

rt

rd

IMM

02

CLK WE

A1
A2
register bank
A3
WD

RD1
RD2
RST

=

MUX

RegWriteE

MemToRegE

RegWriteM

MemToRegM

ALU

ADR
data memory
D

WE
Q

HAZARD UNIT

RegWrite(M) &&
rs(E) == rd(M)

sub $t2, $s0, $s5

or $t1,$s4,$s0

and $t0, $s0, $s1

add $s0, $s2, $s3

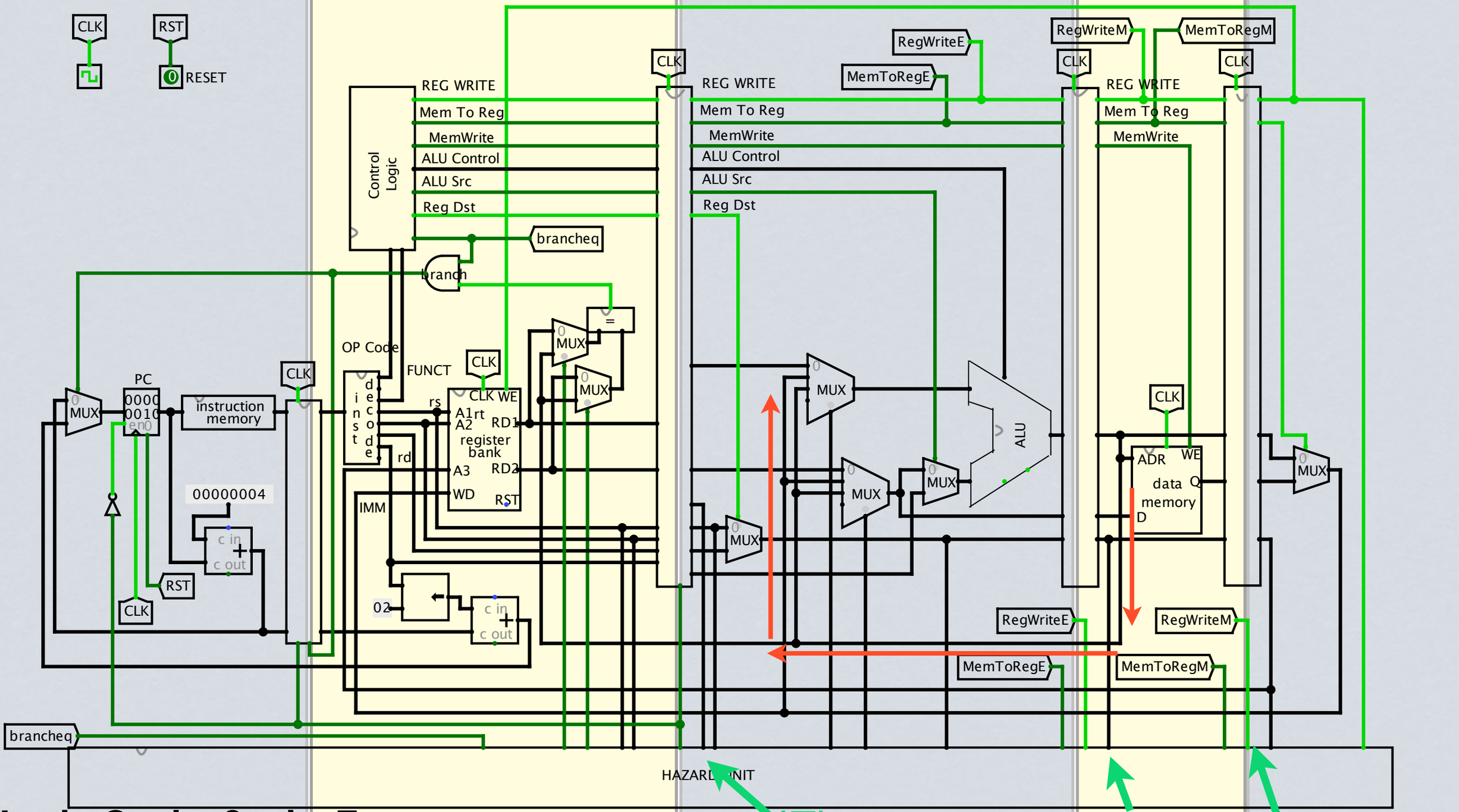
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



sub $t2, $s0, $s5

or $t1,$s4,$s0

and $t0, $s0, $s1

add $s0, $s2, $s3

rs(E)

rd(M)

RegWrite(M)

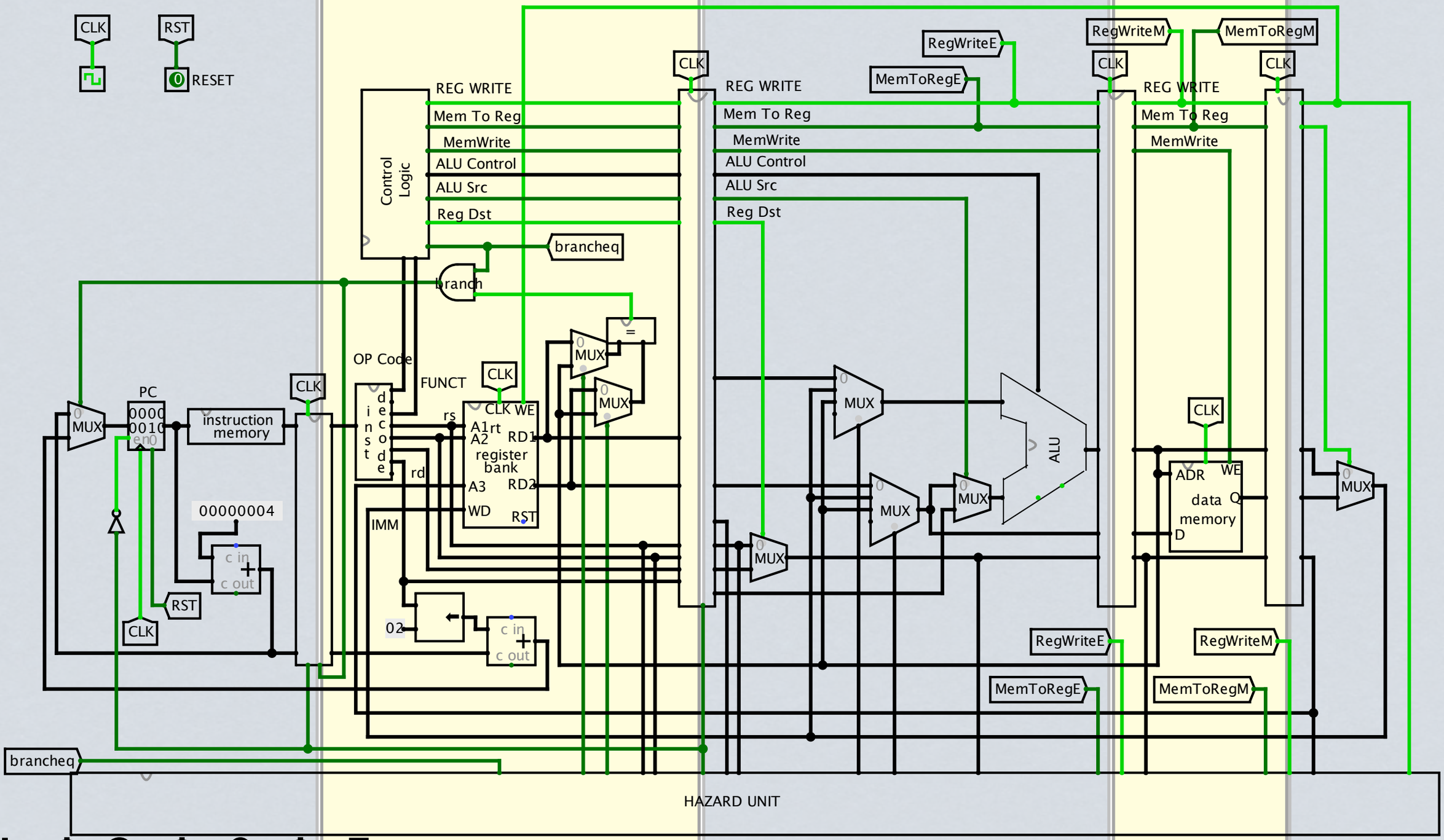
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



sub $t2, $s0, $s5

or $t1,$s4,$s0

and $t0, $s0, $s1

add $s0, $s2, $s3

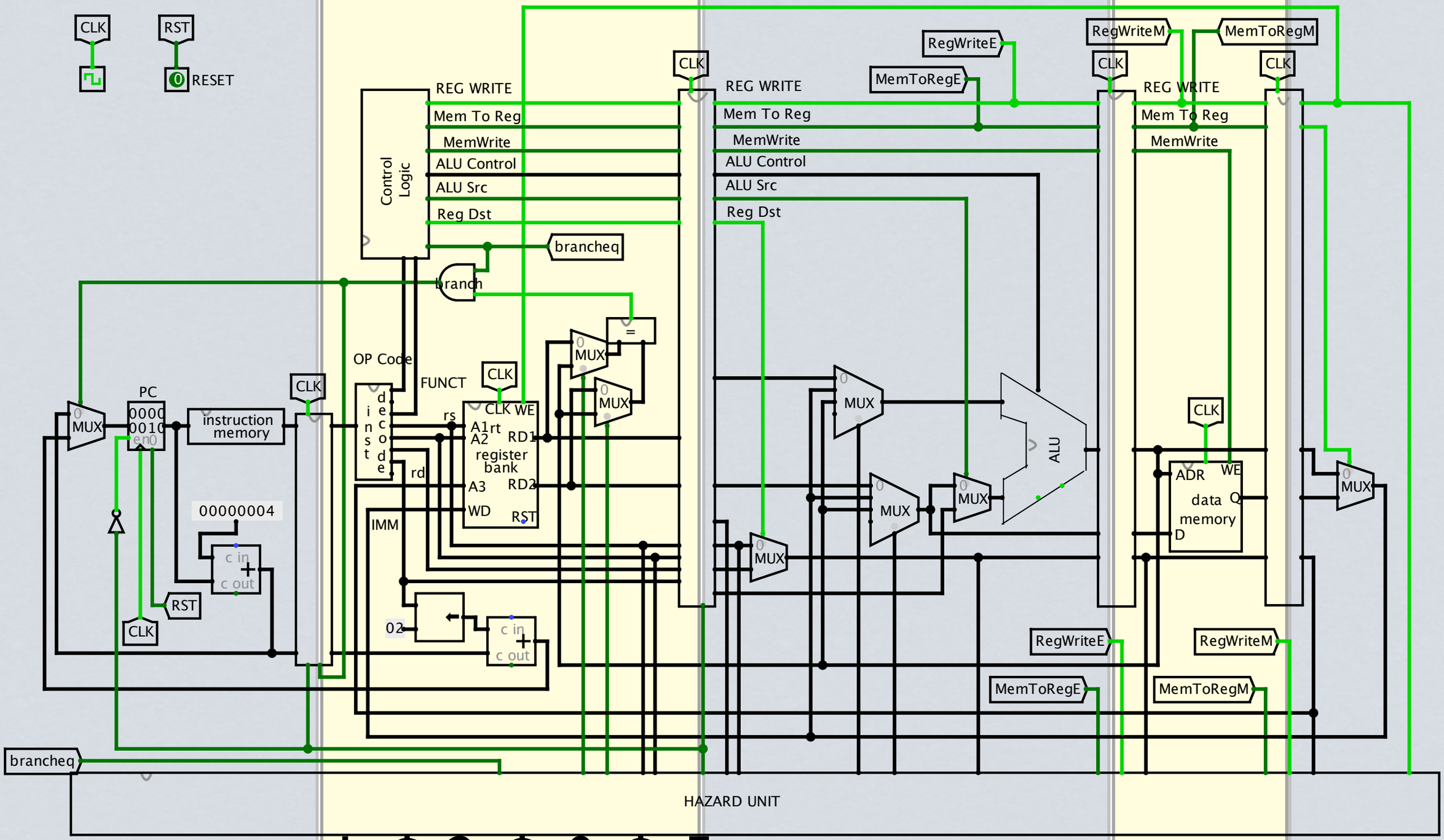
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



sub $t2, $s0, $s5

or $t1,$s4,$s0

and $t0, $s0, $s1

add $s0, $s2

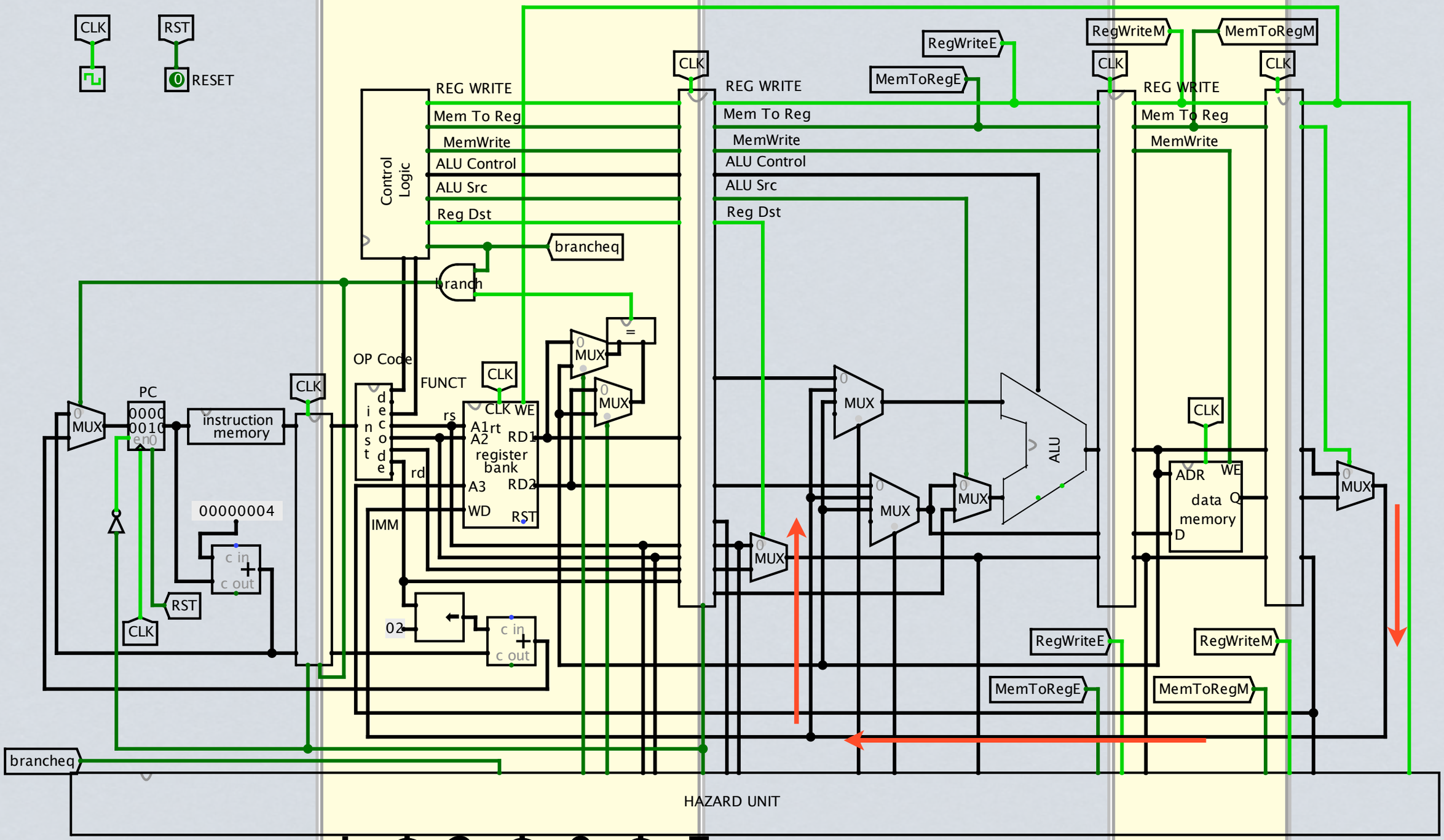
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



sub $t2, $s0, $s5

or $t1,$s4,$s0

and $t0, $s0, $s1

add $s0, $s2

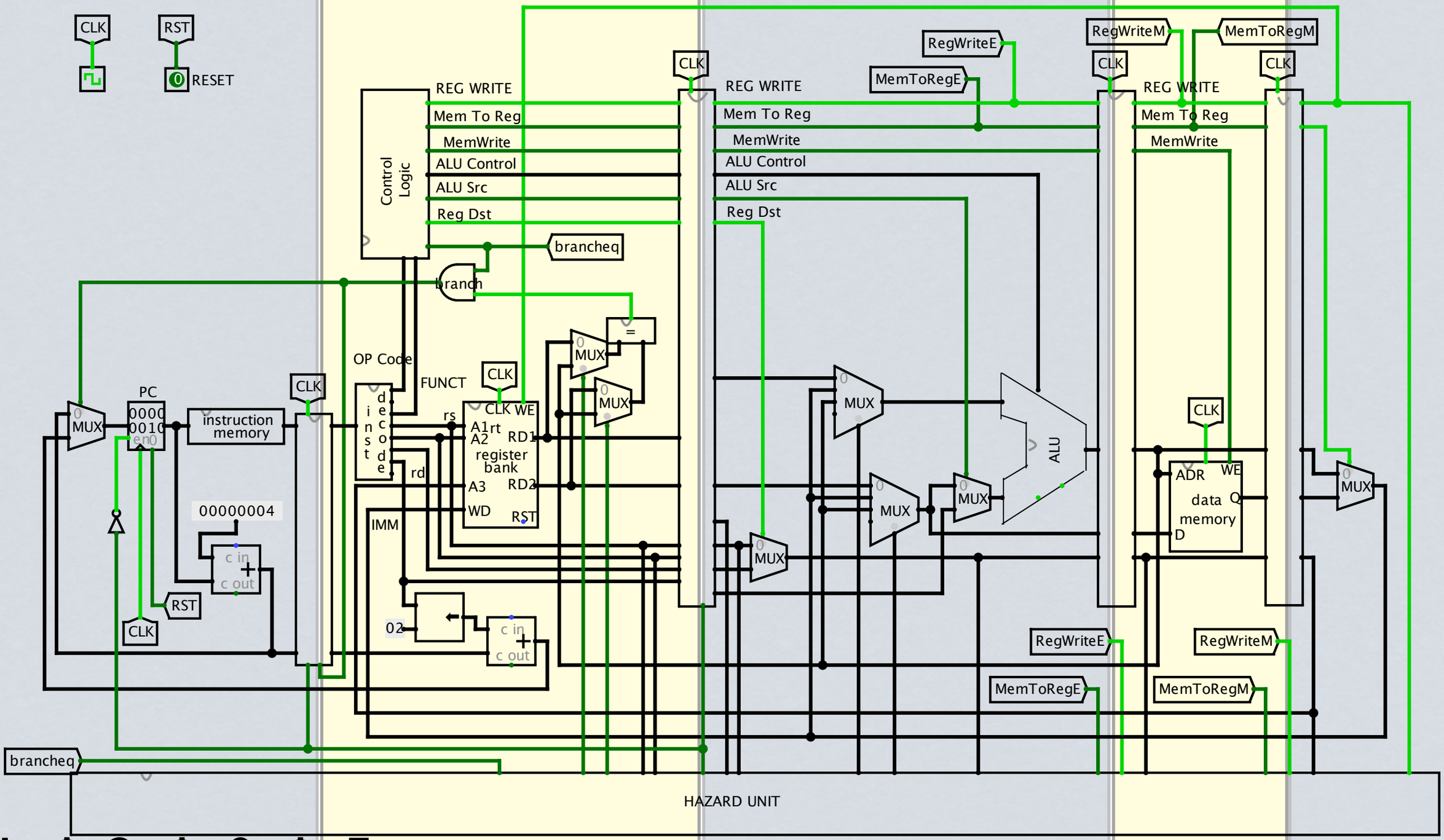
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



sub $t2, $s0, $s5

or $t1,$s4,$s0

and $t0, $s0, $s1

add $s0, $s2, $s3

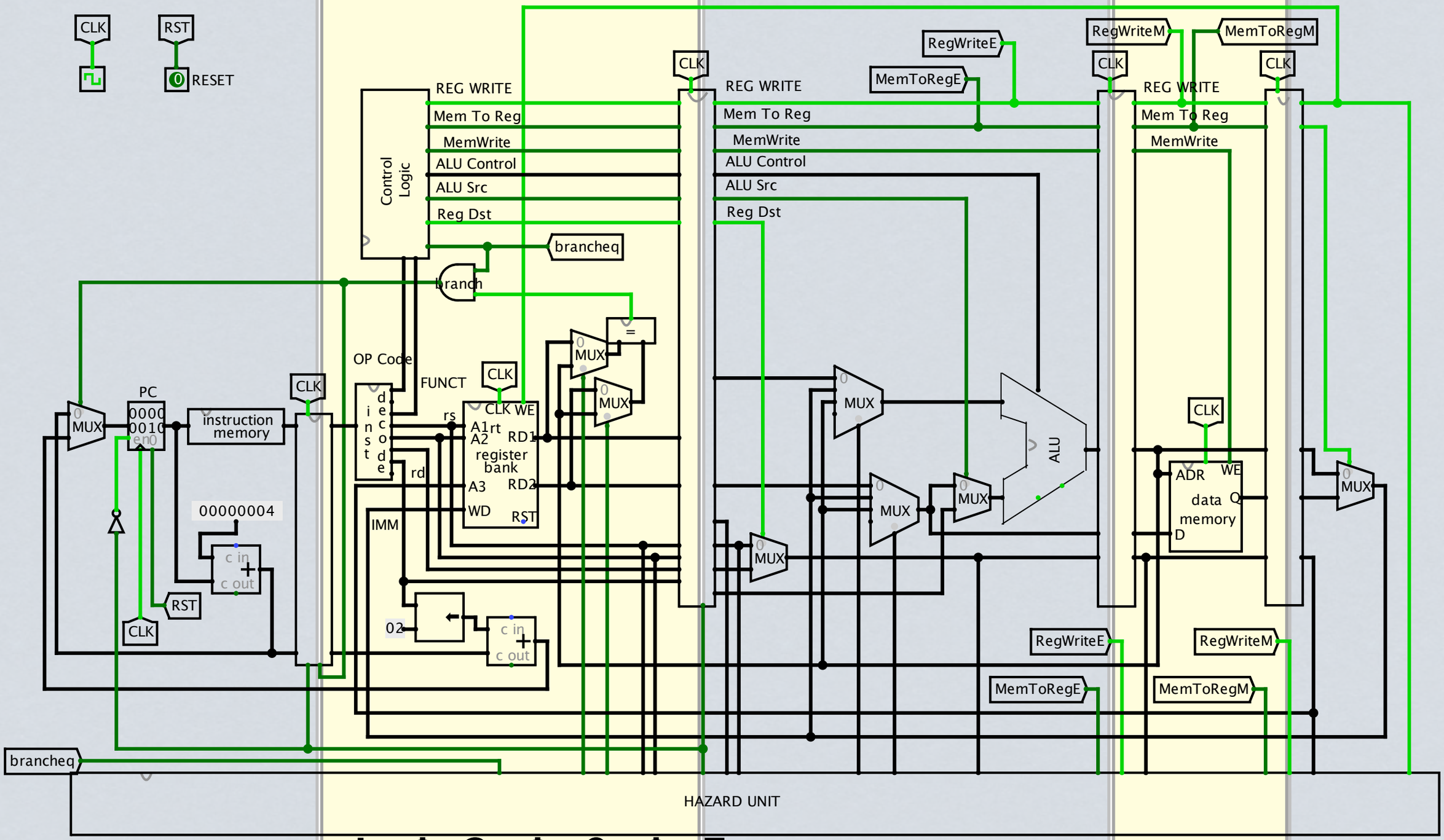
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



sub $t2, $s0, $s5

or $t1,$s4,$s0

and $t0, $s0, $s1

add $s0, $s2,

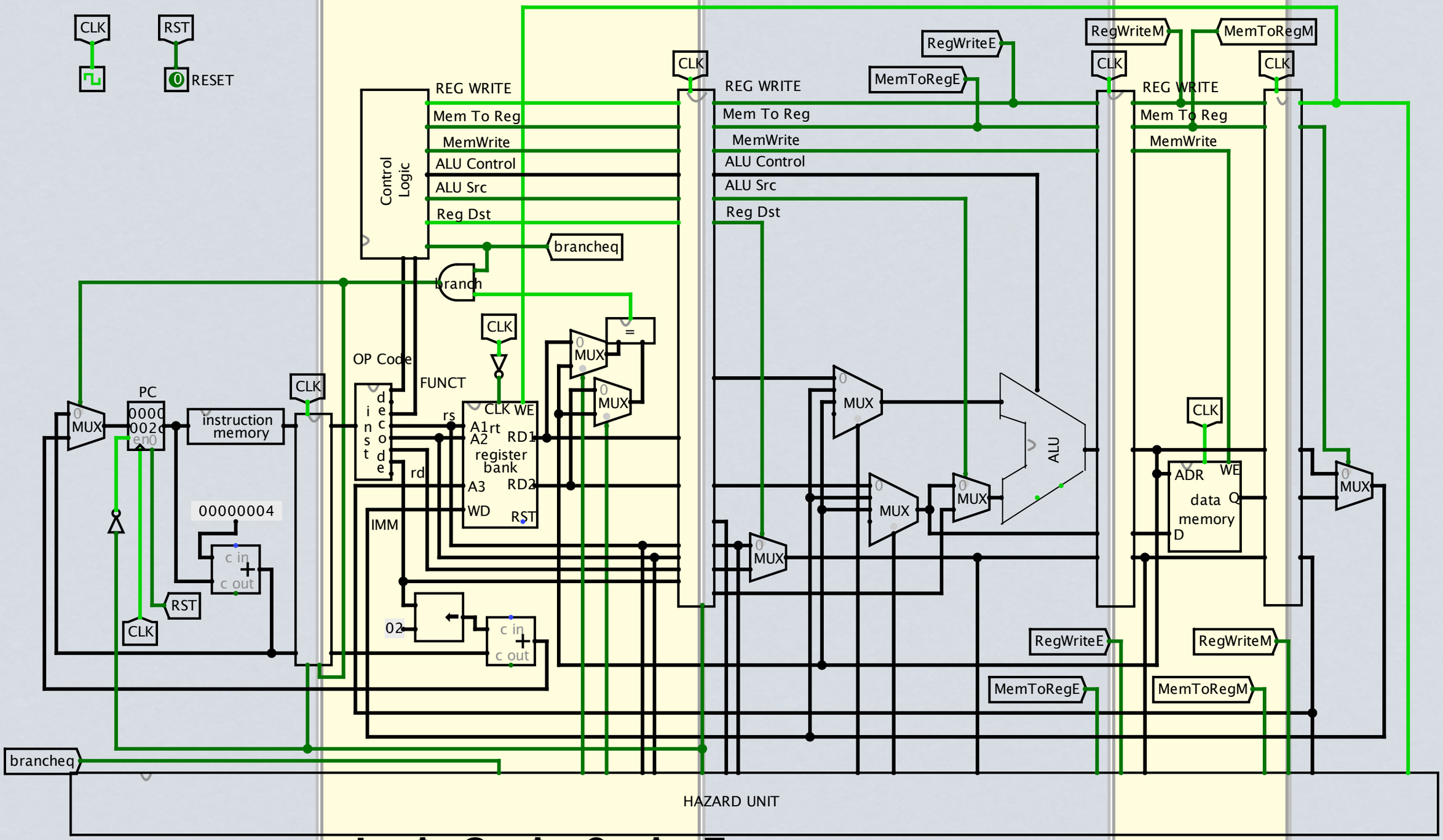
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



sub $t2, $s0, $s5

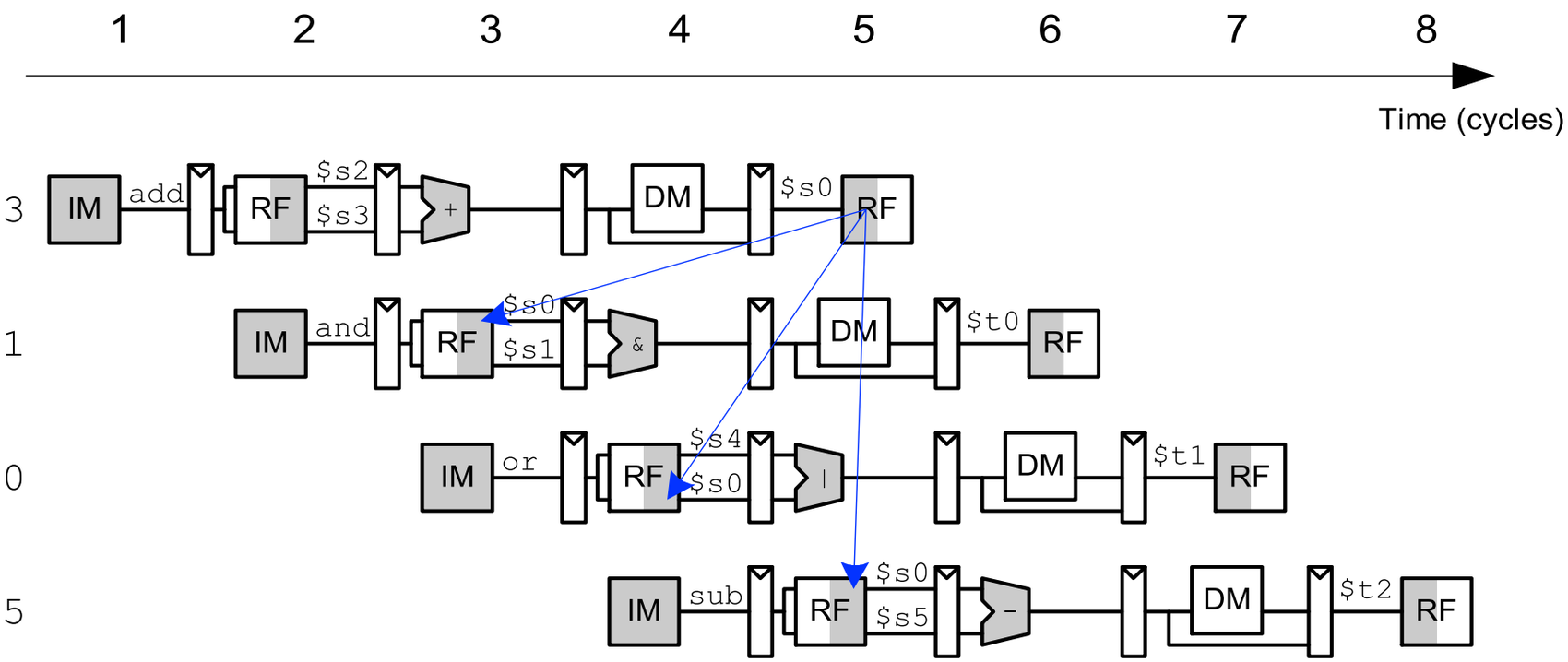
or $t1,$s4,$s0

and $t0, $s0, $s1

add $s0, $s2,

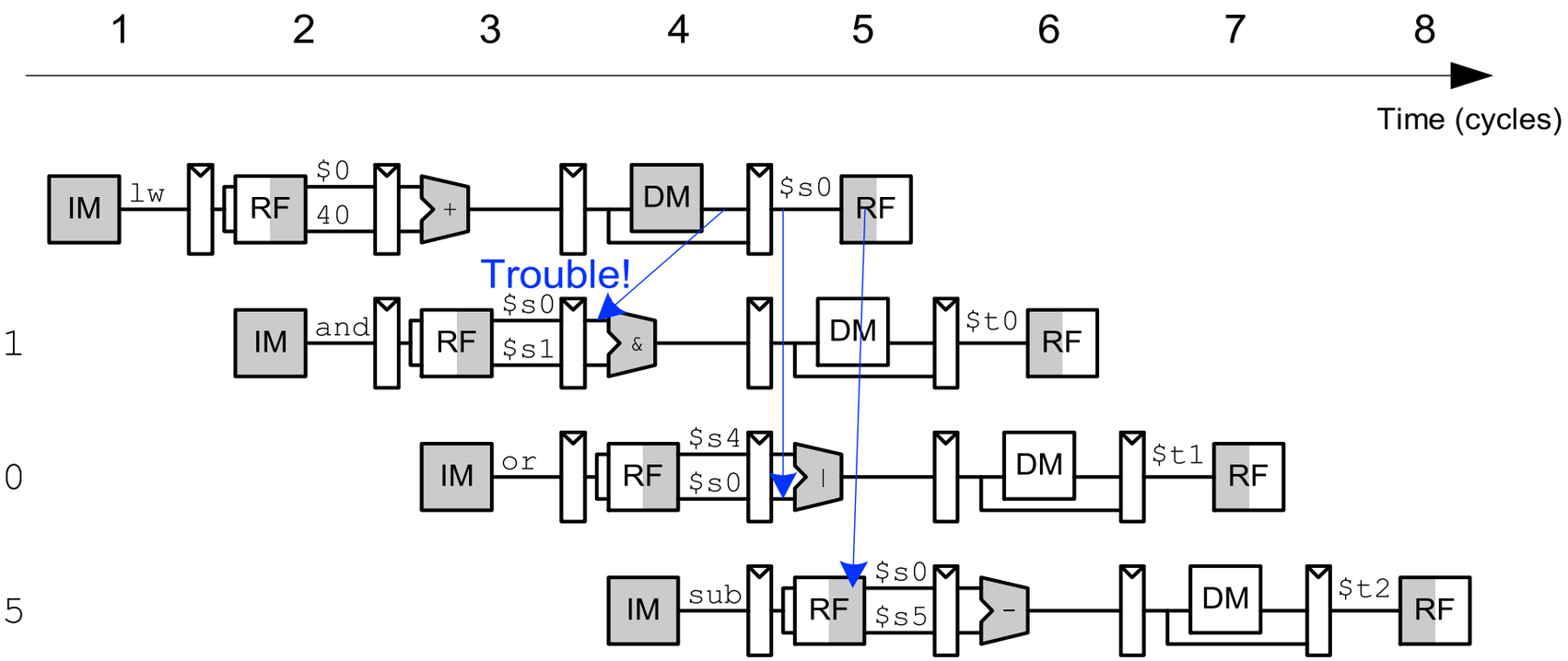
Data Hazards and Forwarding

```
add $s0, $s2, $s3

and $t0, $s0, $s1

or  $t1, $s4, $s0

sub $t2, $s0, $s5
```



Stalling

```
lw $s0, 40($0)

and $t0, $s0, $s1

or  $t1, $s4, $s0

sub $t2, $s0, $s5
```



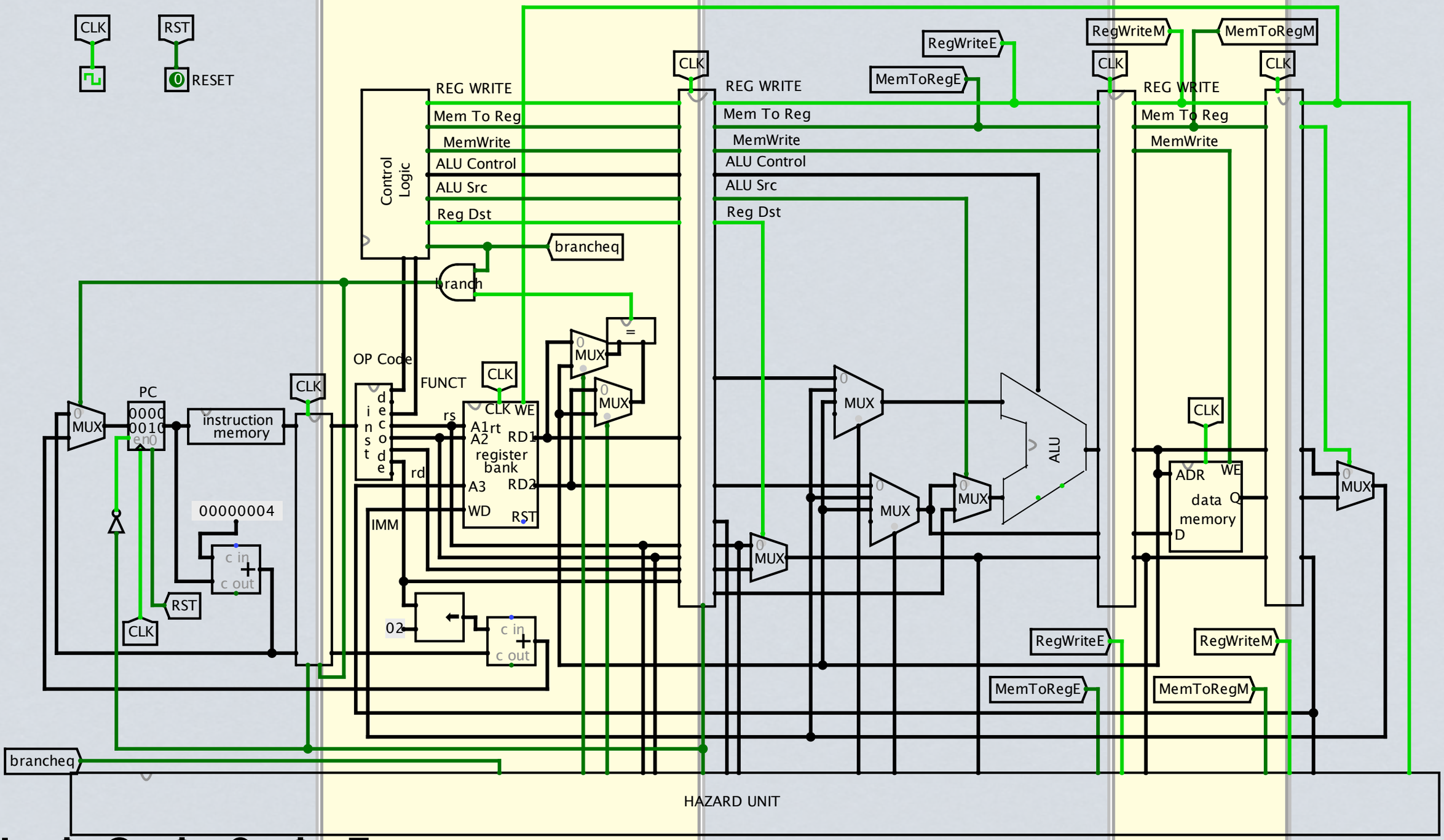
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



sub $t2, $s0, $s5

or $t1,$s4,$s0

and $t0, $s0, $s1

lw $s0, 40($0)

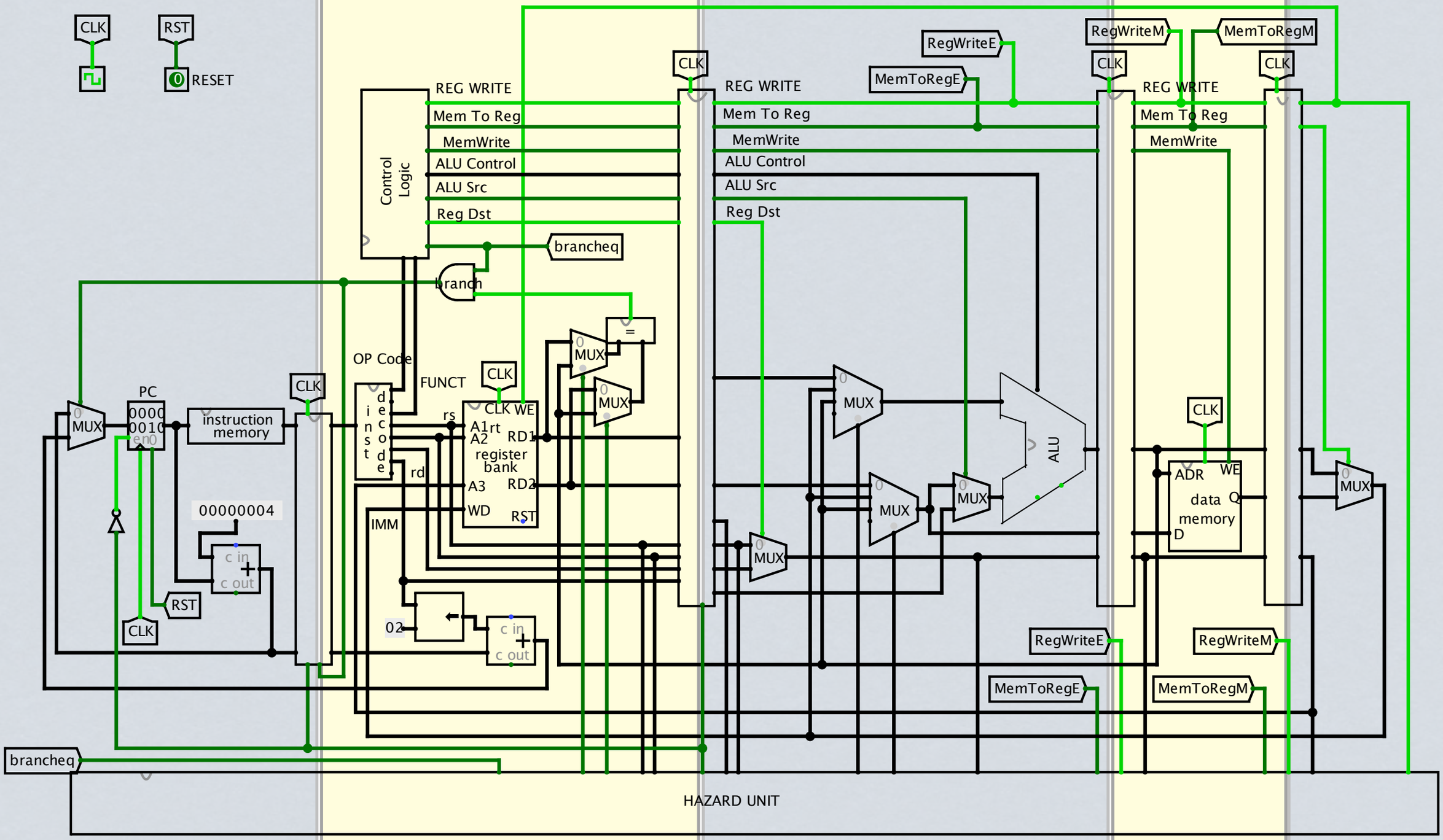
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



or $t1,$s4,$s0

and $t0, $s0, $s1

lw $s0, 40($0)