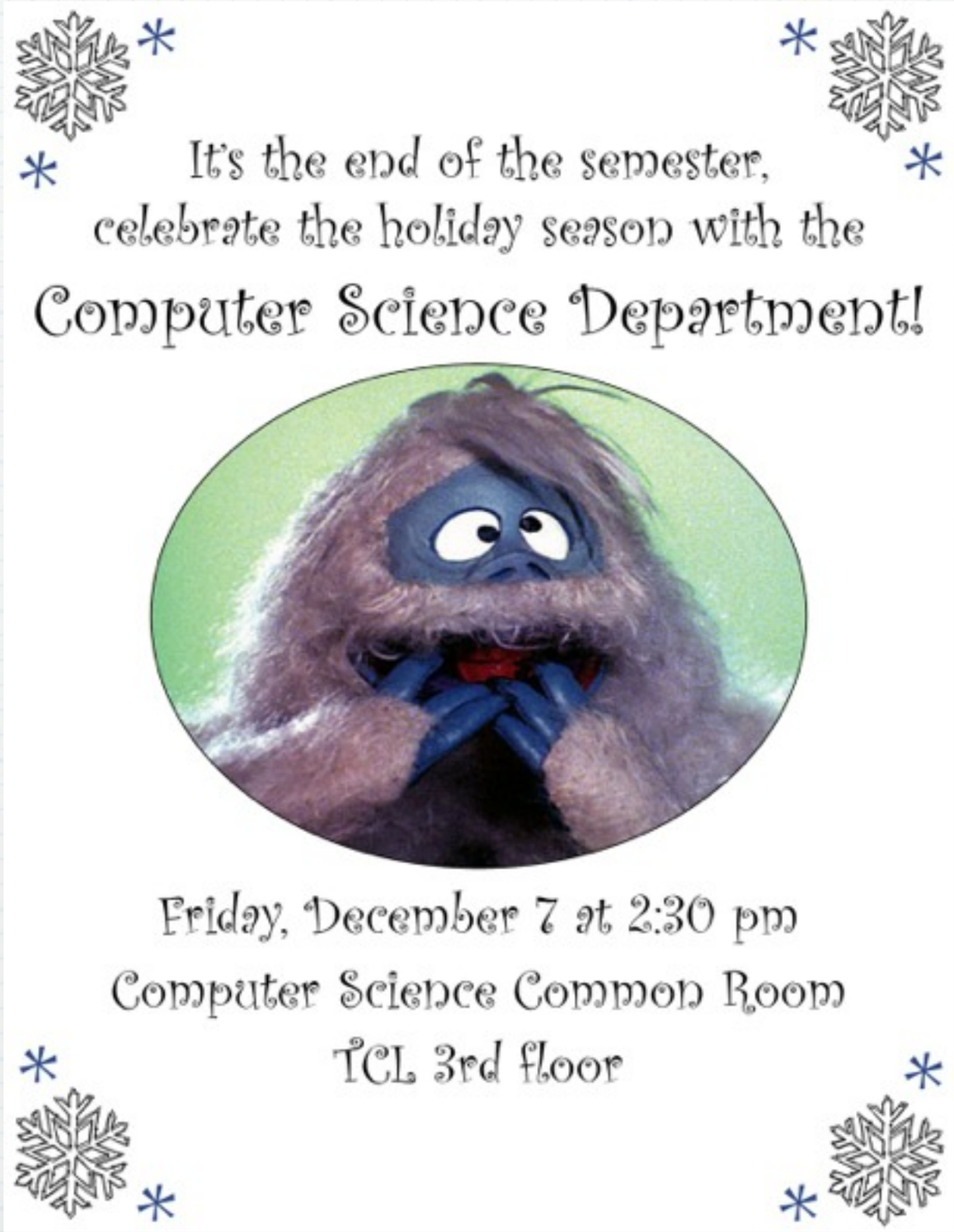
Announcements

Final projects due 11pm, Thursday



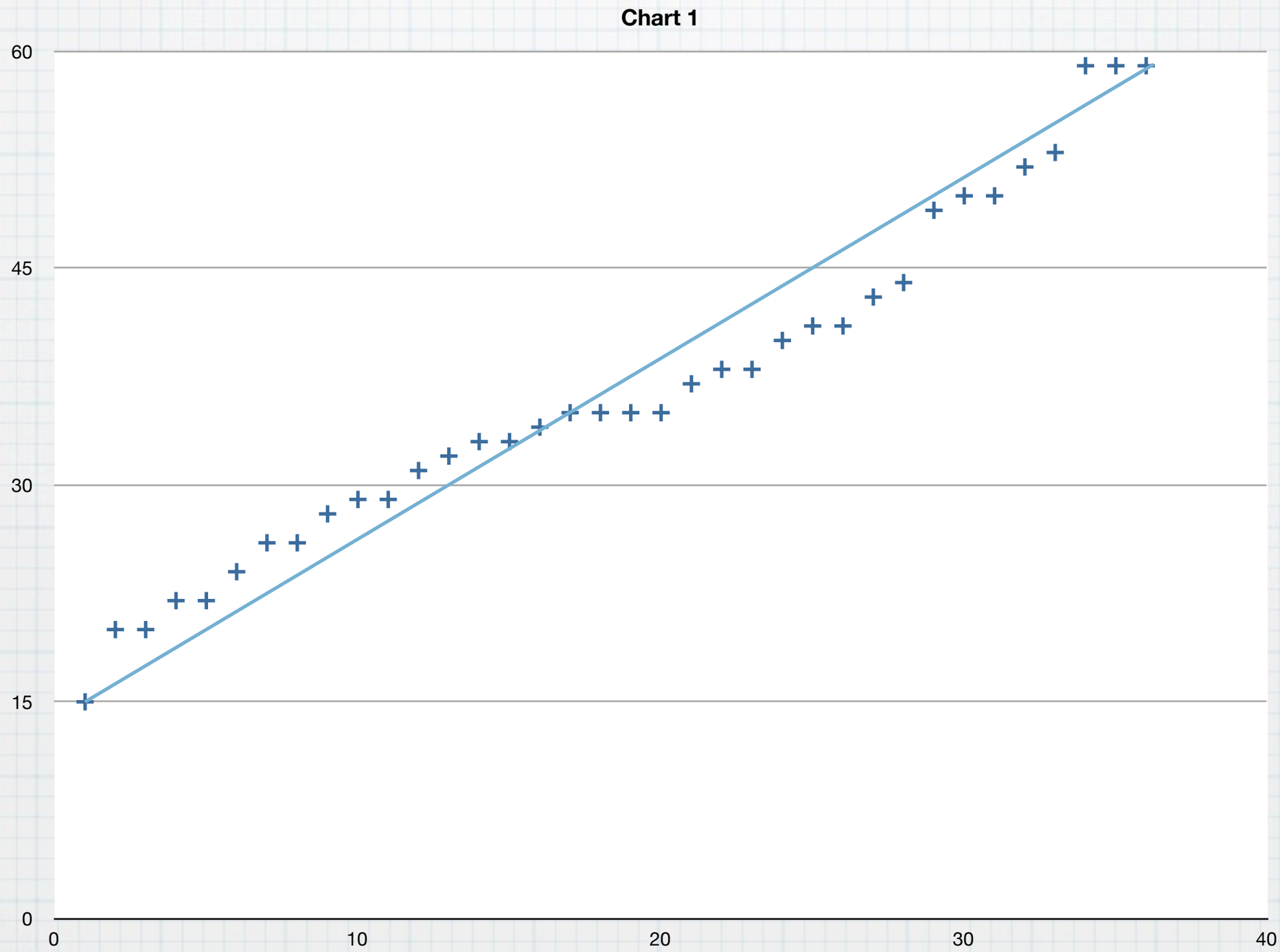


Chart 1
60
45
30
15
0
0
10
20
30
40

Chart 1

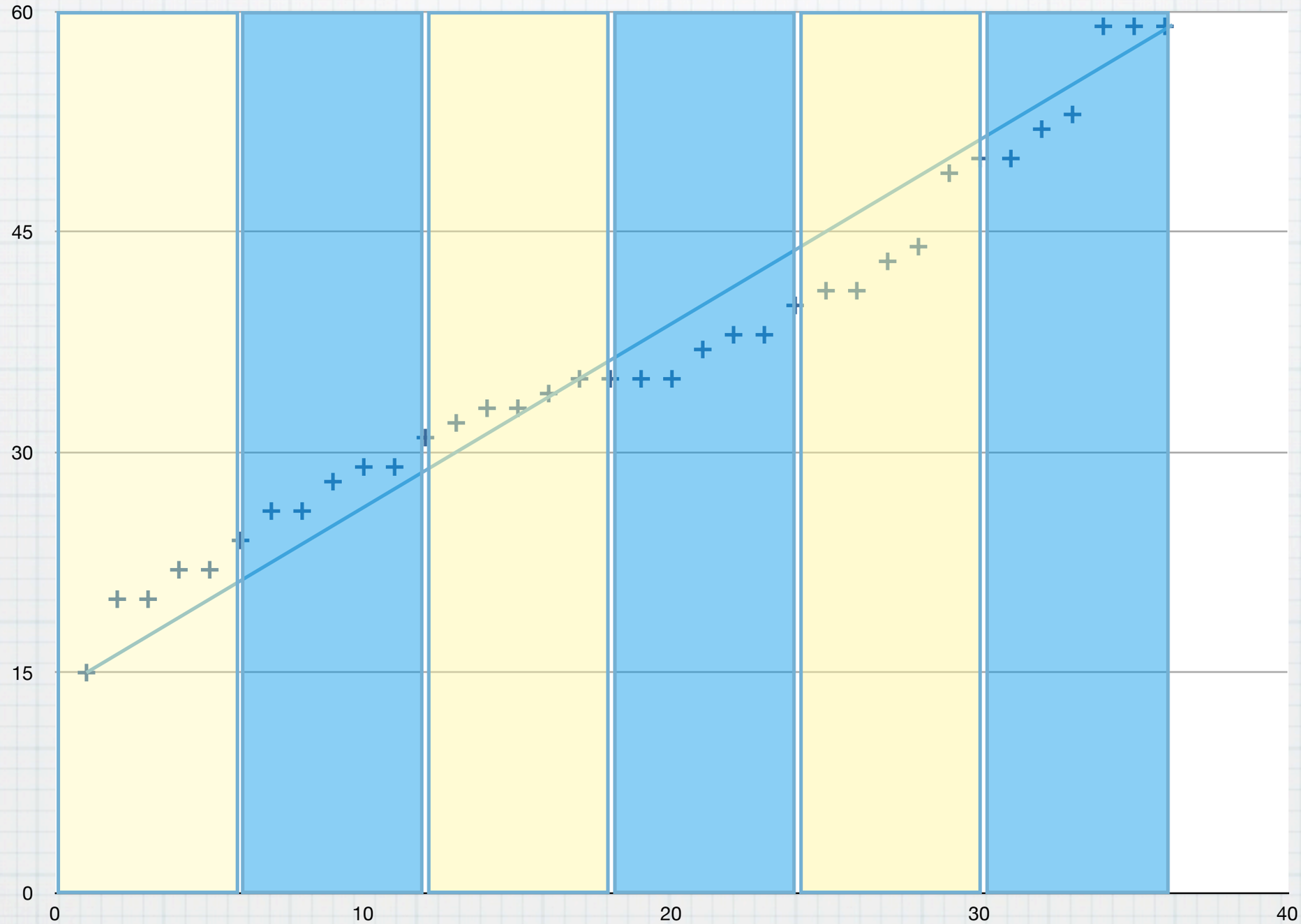
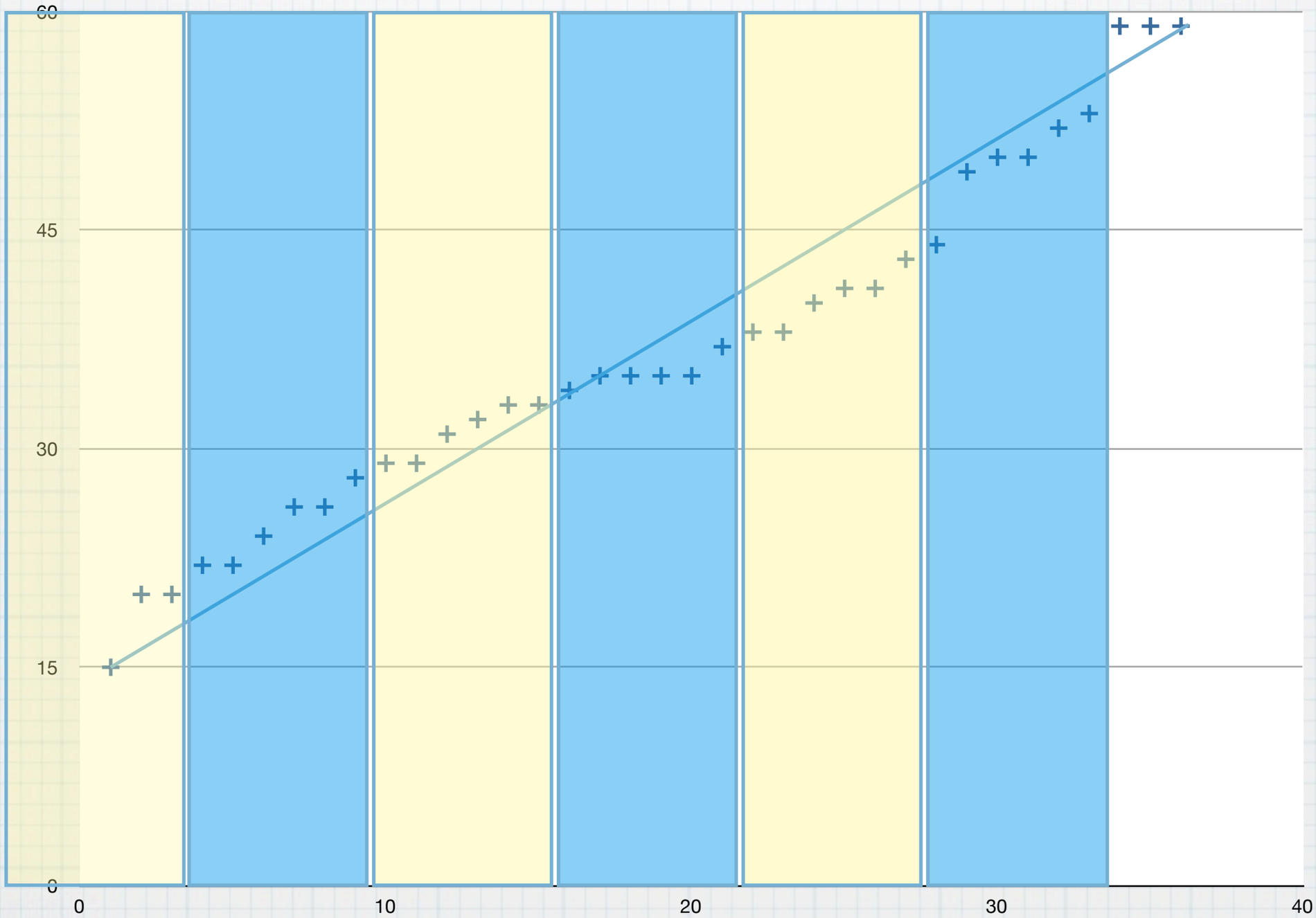
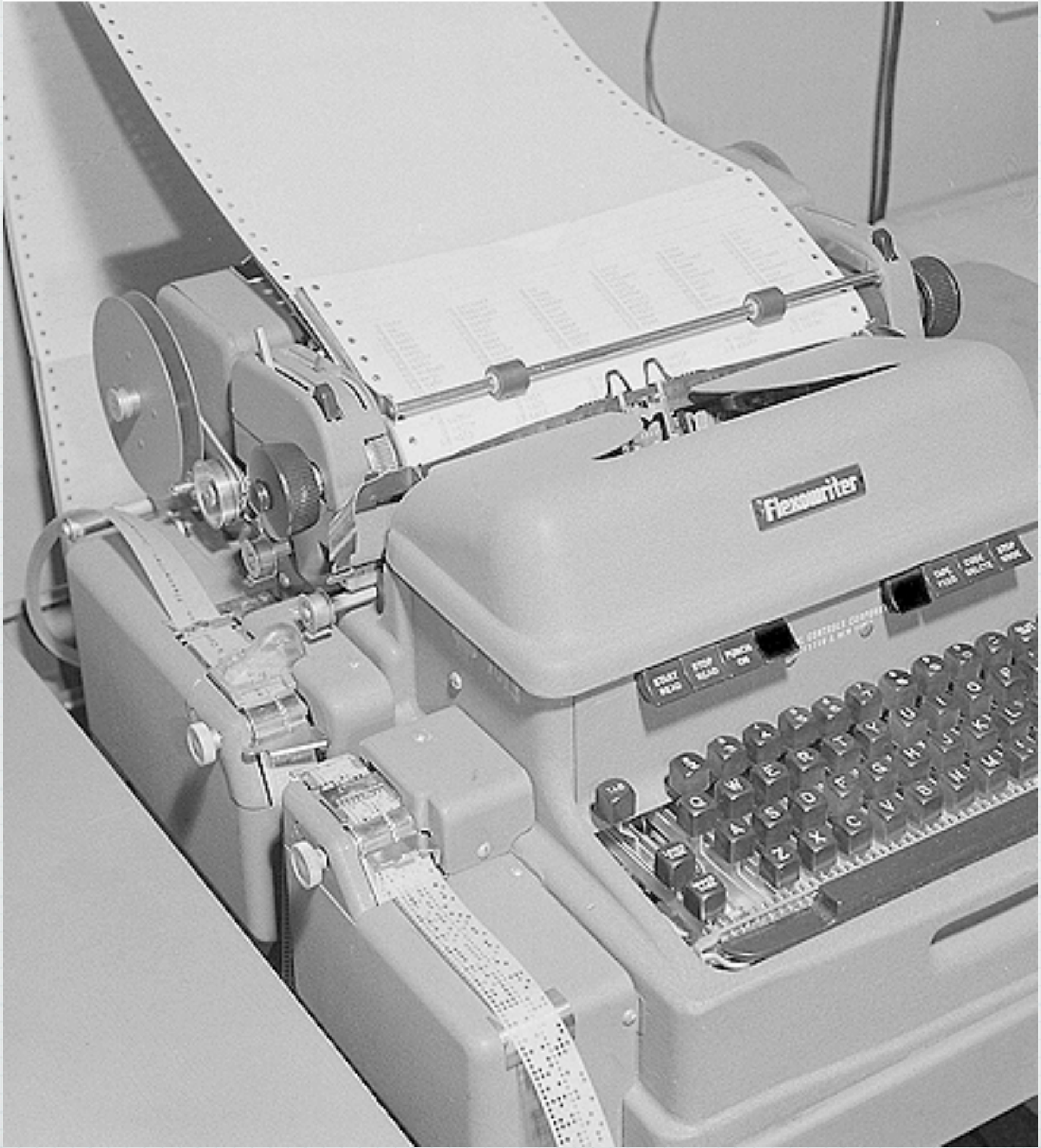
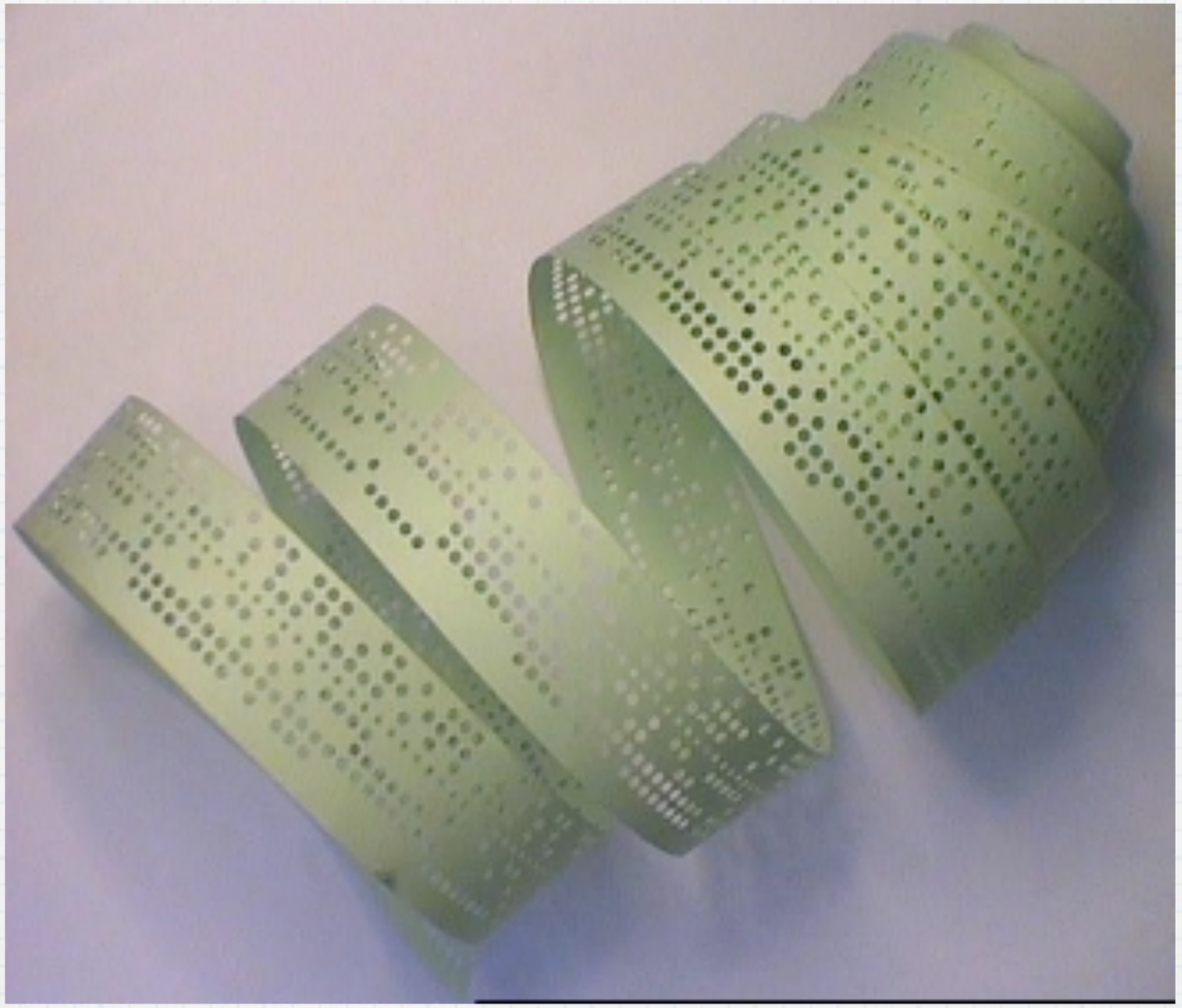


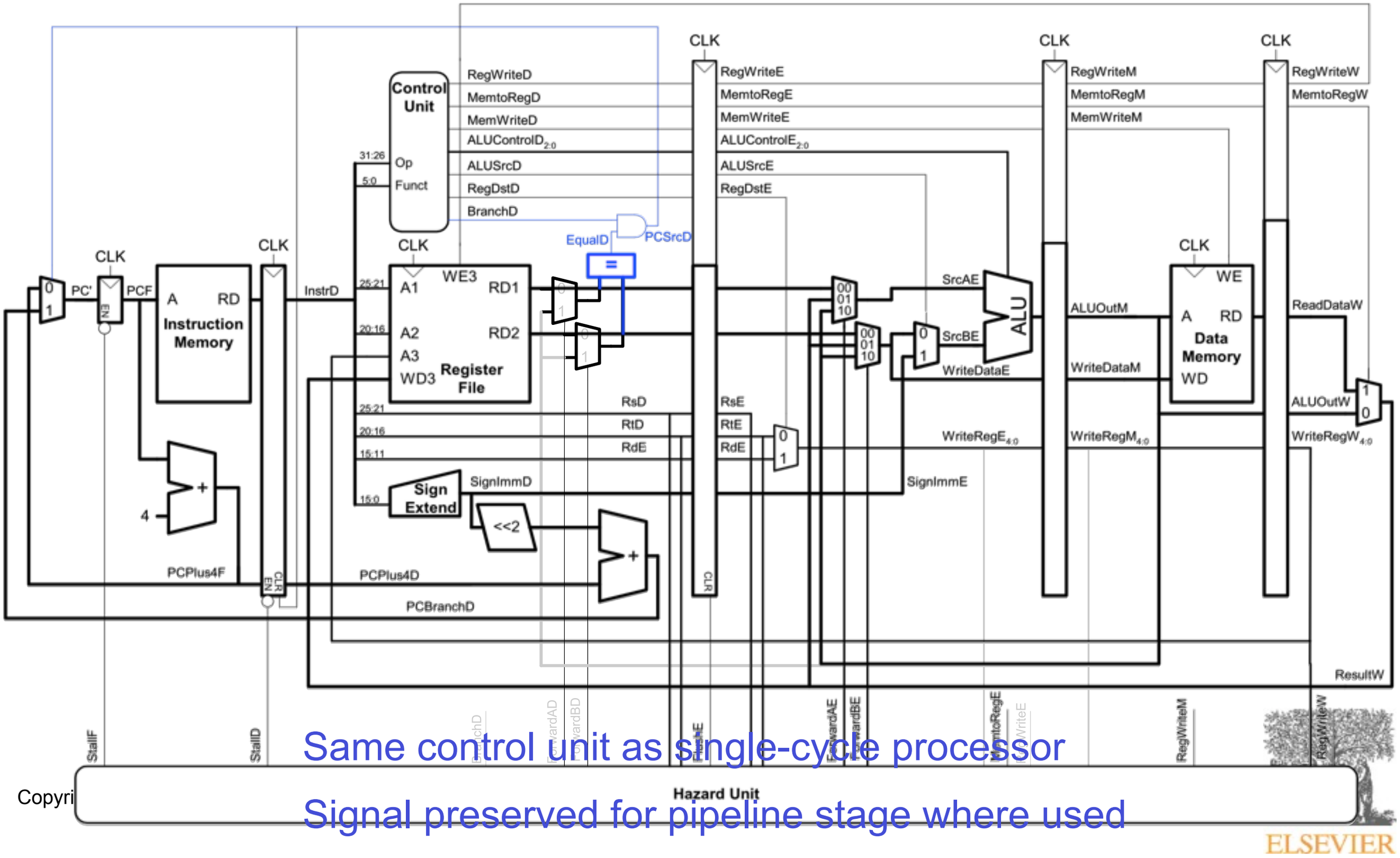
Chart 1




Flexowriter



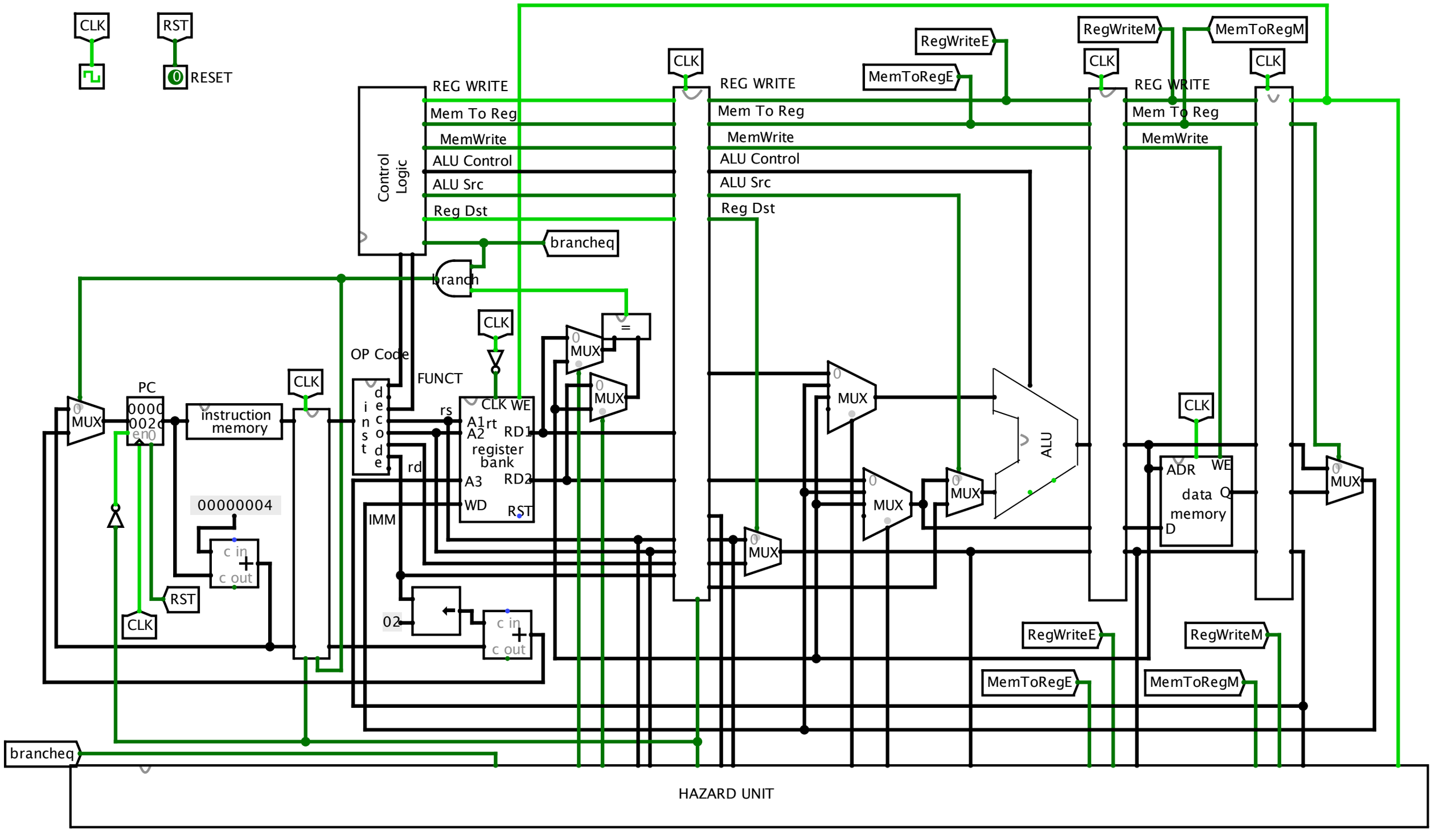
Pipelined Processor



Same control unit as single-cycle processor

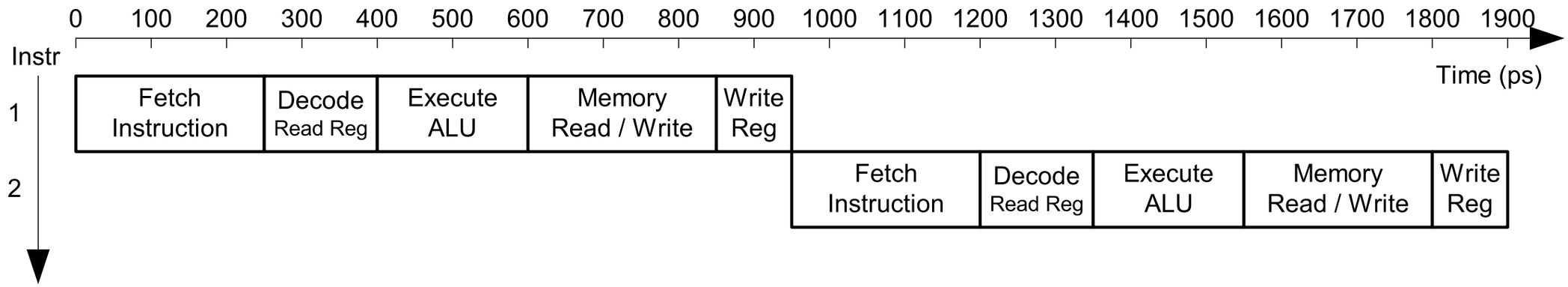
Signal preserved for pipeline stage where used

Pipelining

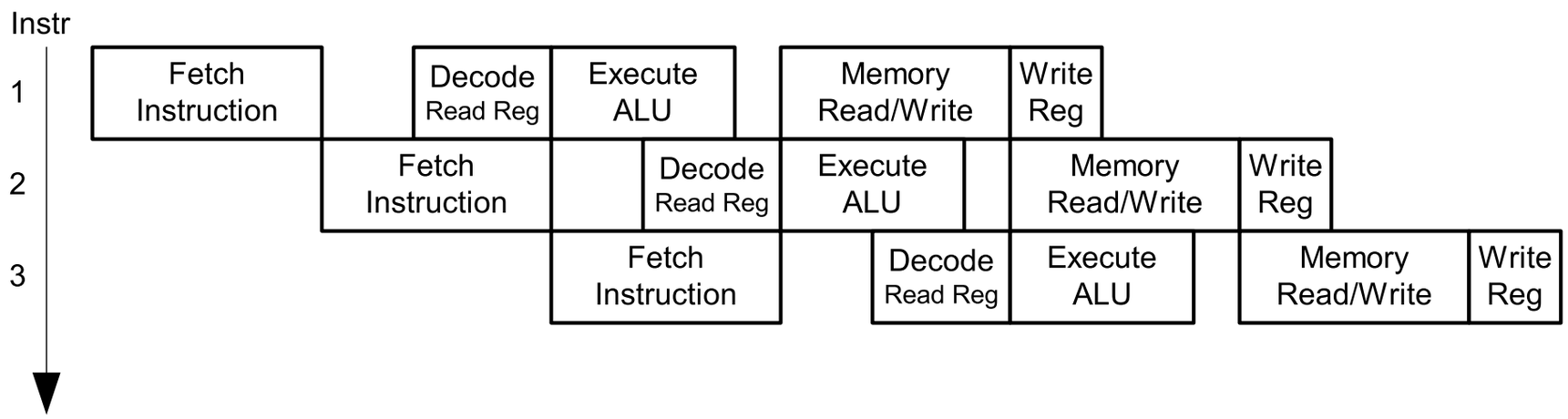

CLK
RST
RESET
REG WRITE
Mem To Reg
MemWrite
ALU Control
ALU Src
Reg Dst
Control Logic
brancheq
branch
CLK
REG WRITE
Mem To Reg
MemWrite
ALU Control
ALU Src
Reg Dst
RegWriteE
MemToRegE
RegWriteM
MemToRegM
CLK
REG WRITE
Mem To Reg
MemWrite
CLK
OP Code
FUNCT
PC
0000 0020
en0
instruction memory
inst
decode
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

Single-Cycle vs. Pipelined Performance

Single-Cycle

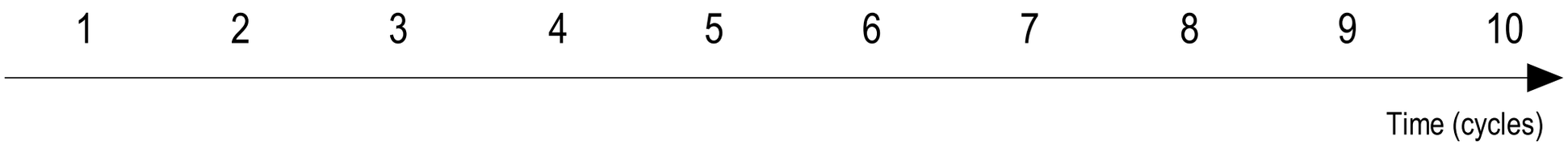


Pipelined

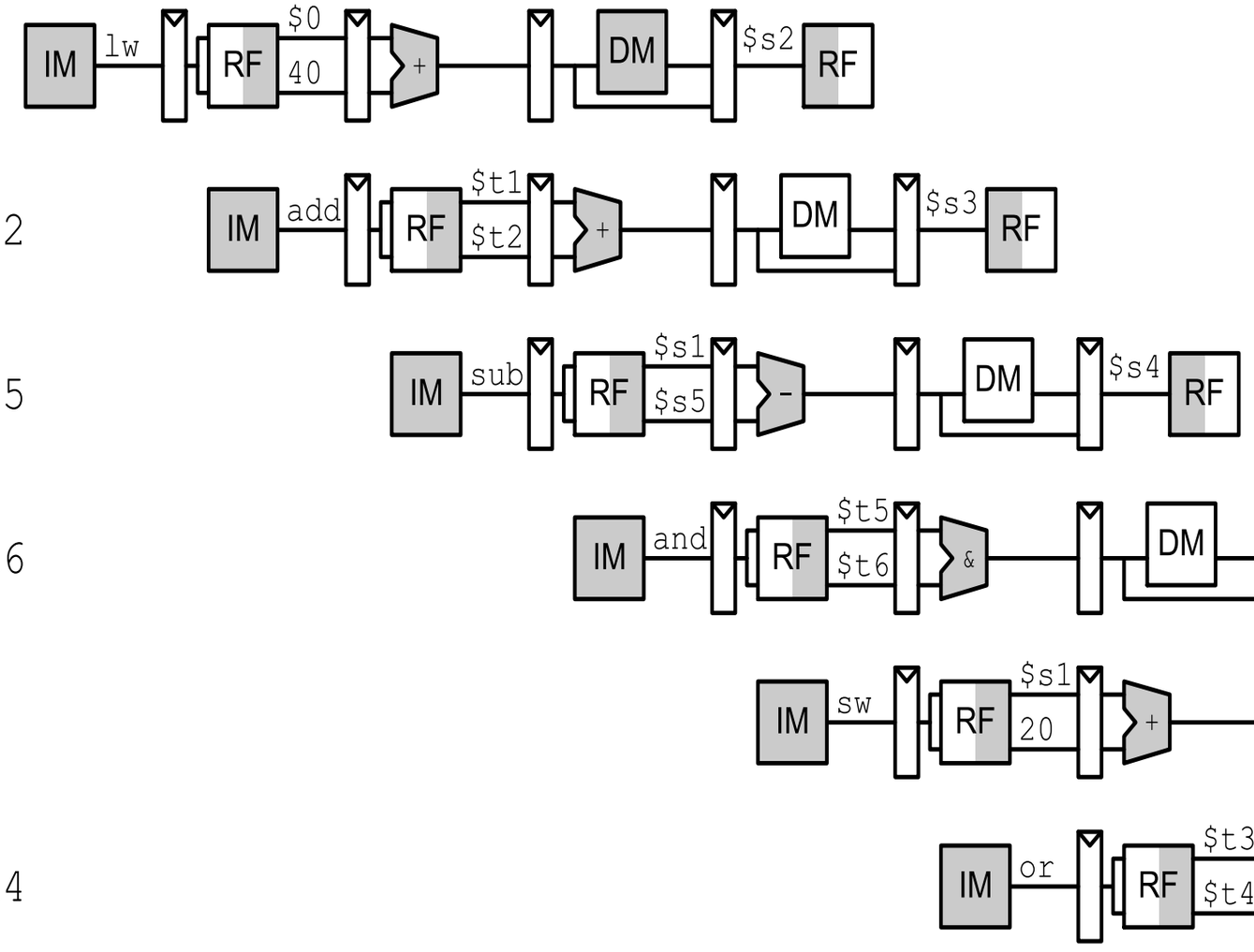


ELSEVIER

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
```



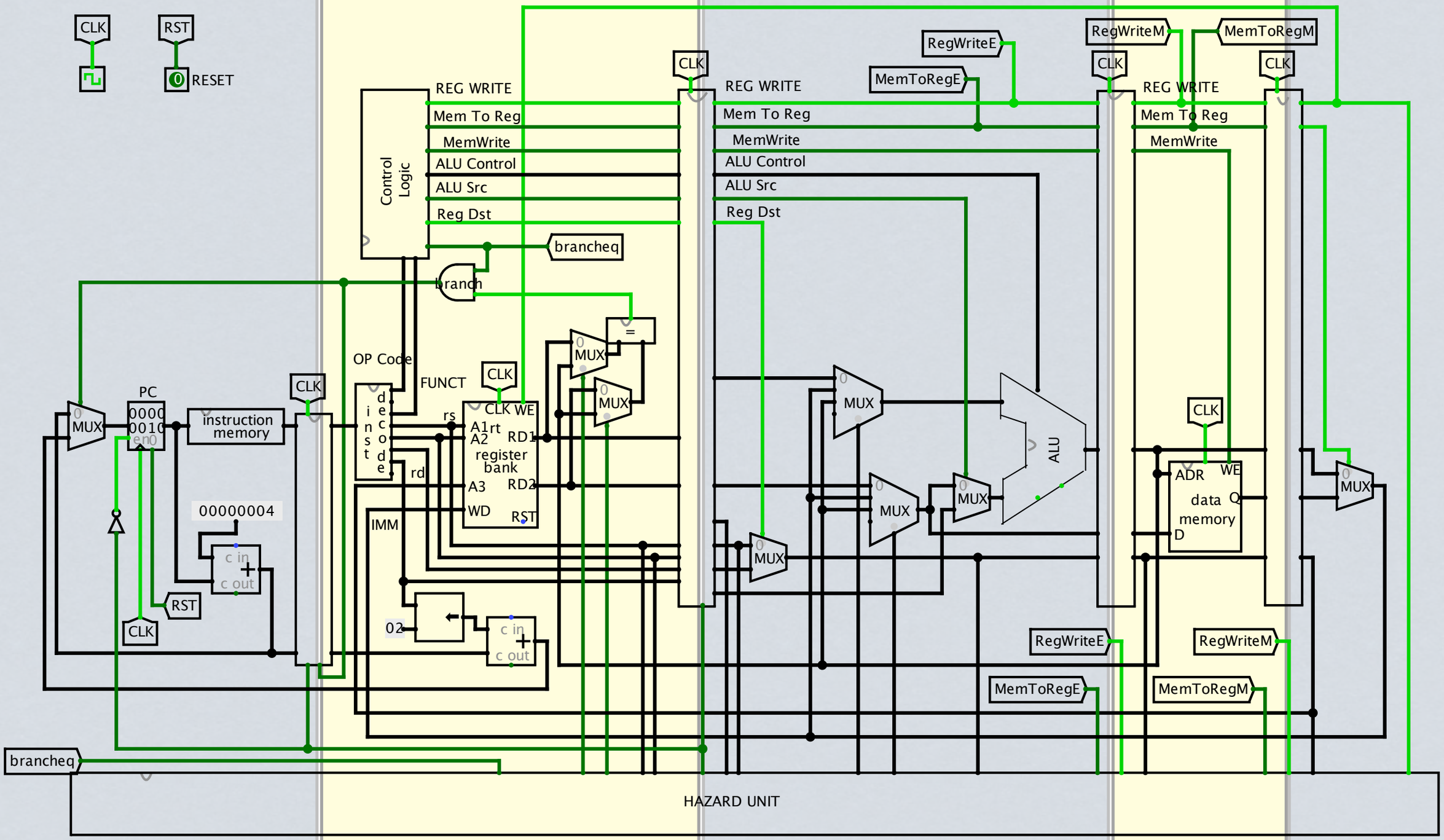
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



sw $s6, 20($s1)

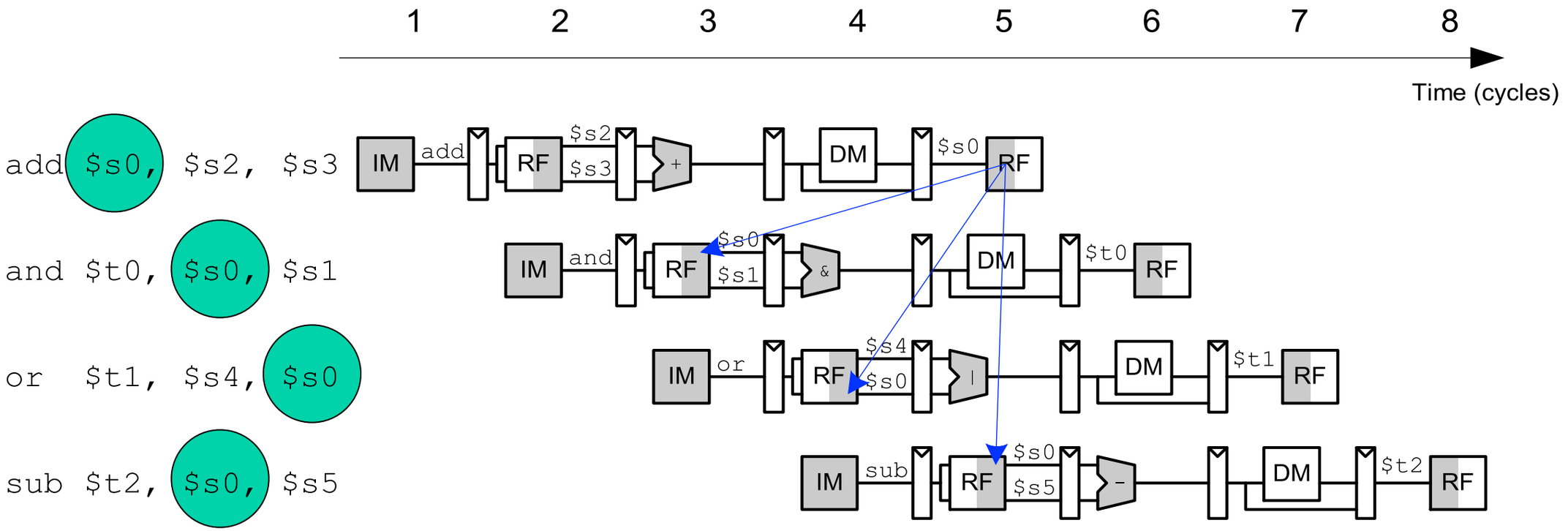
and $s5, $t5, $t6

sub $s4, $s1, $s5

add $s3, $t1, $t2

lw $s2,40($0)

Data Hazards and Forwarding



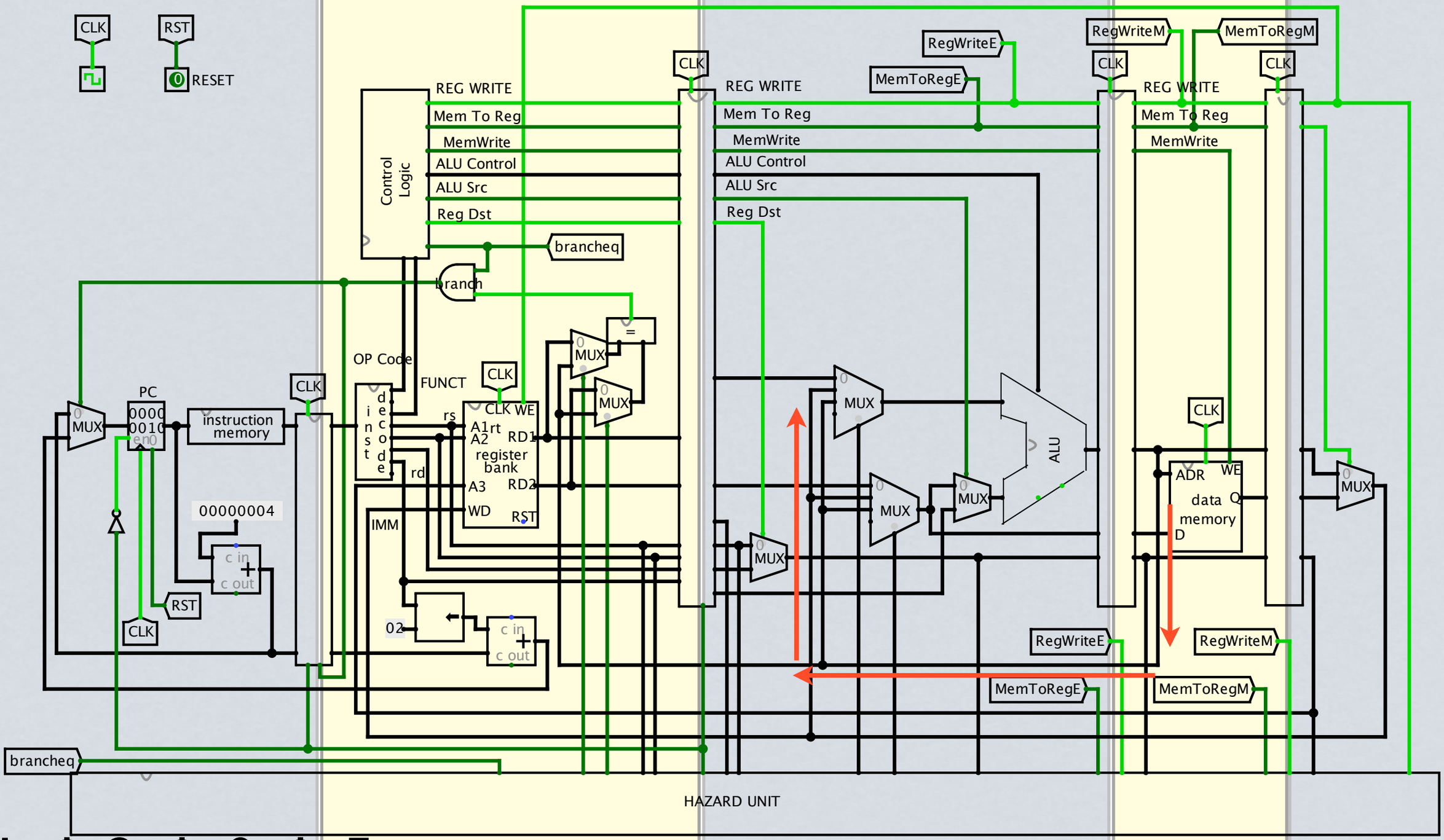
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



sub $t2, $s0, $s5

or $t1,$s4,$s0

and $t0, $s0, $s1

add $s0, $s2, $s3

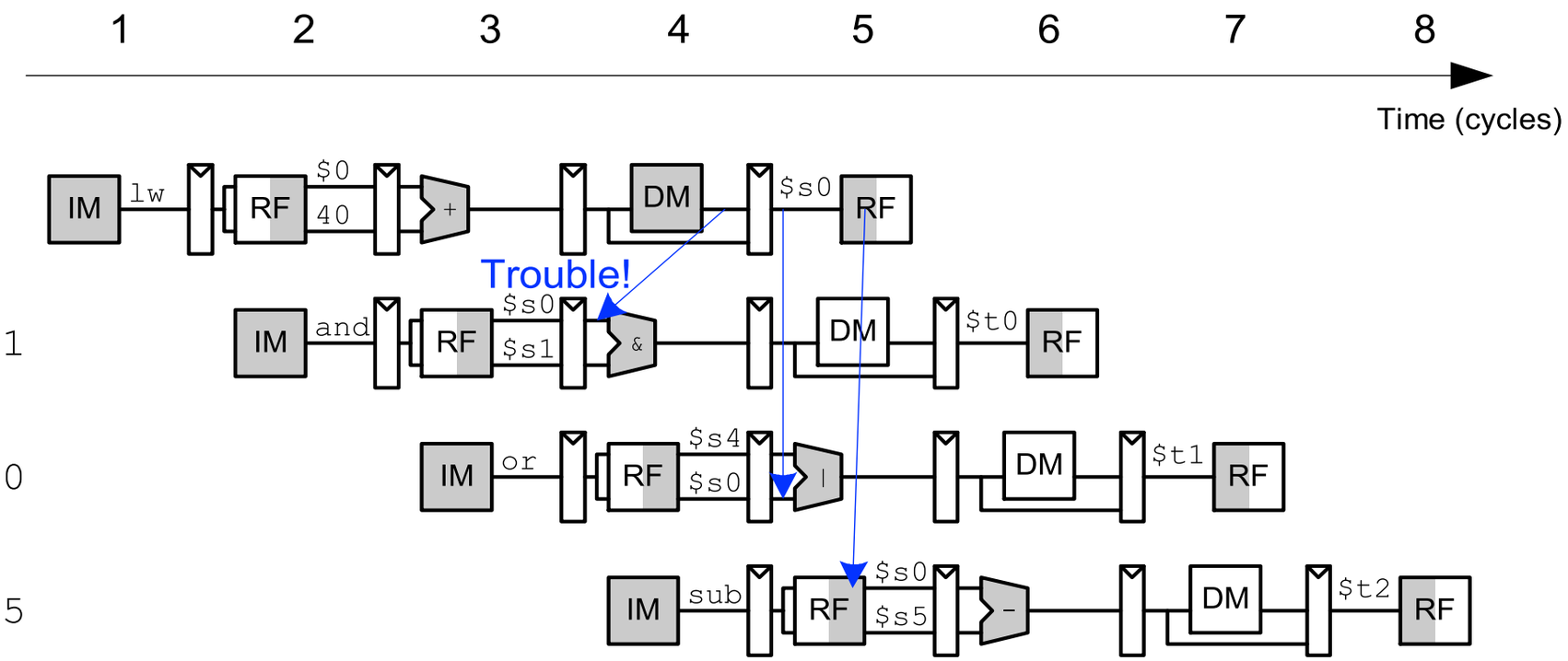
Stalling

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

and $t0, $s0, $s1

or  $t1, $s4, $s0

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



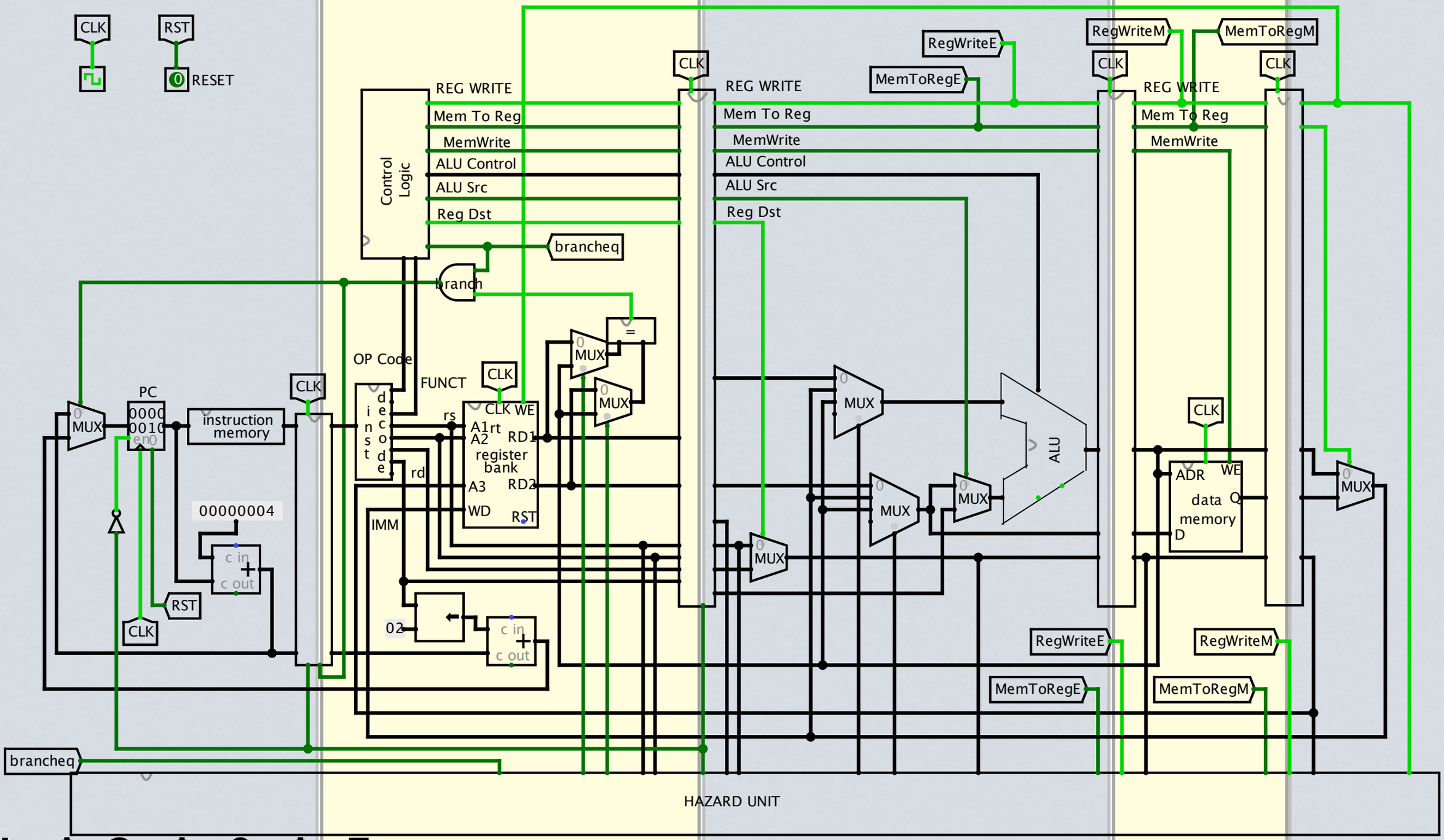
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



sub $t2, $s0, $s5

or $t1,$s4,$s0

and $t0, $s0, $s1

lw $s0, 40($0)

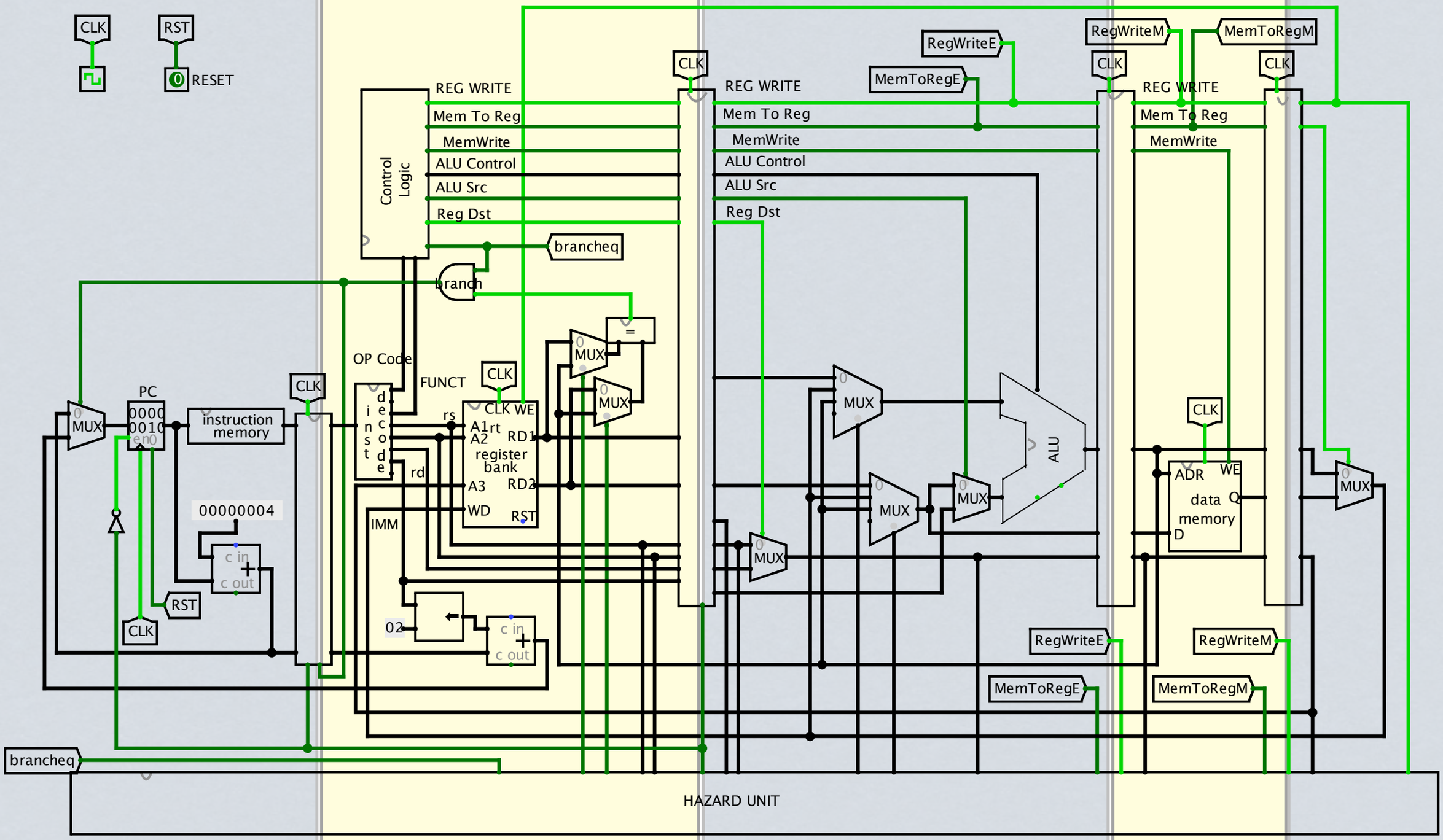
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



or $t1,$s4,$s0

and $t0, $s0, $s1

lw $s0, 40($0)

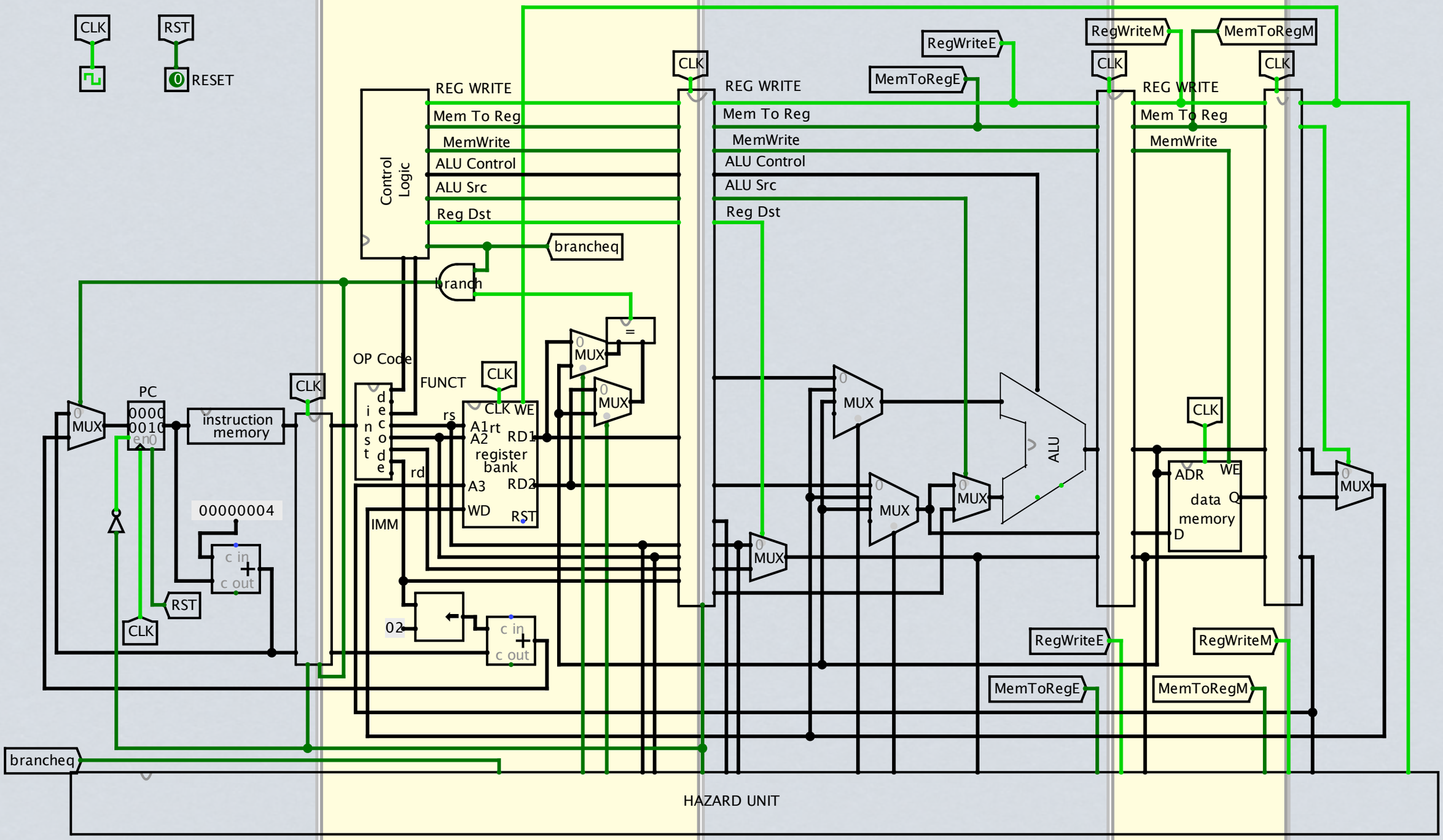
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



or $t1,$s4,$s0

and $t0, $s0, $s1

lw $s0, 40($0)

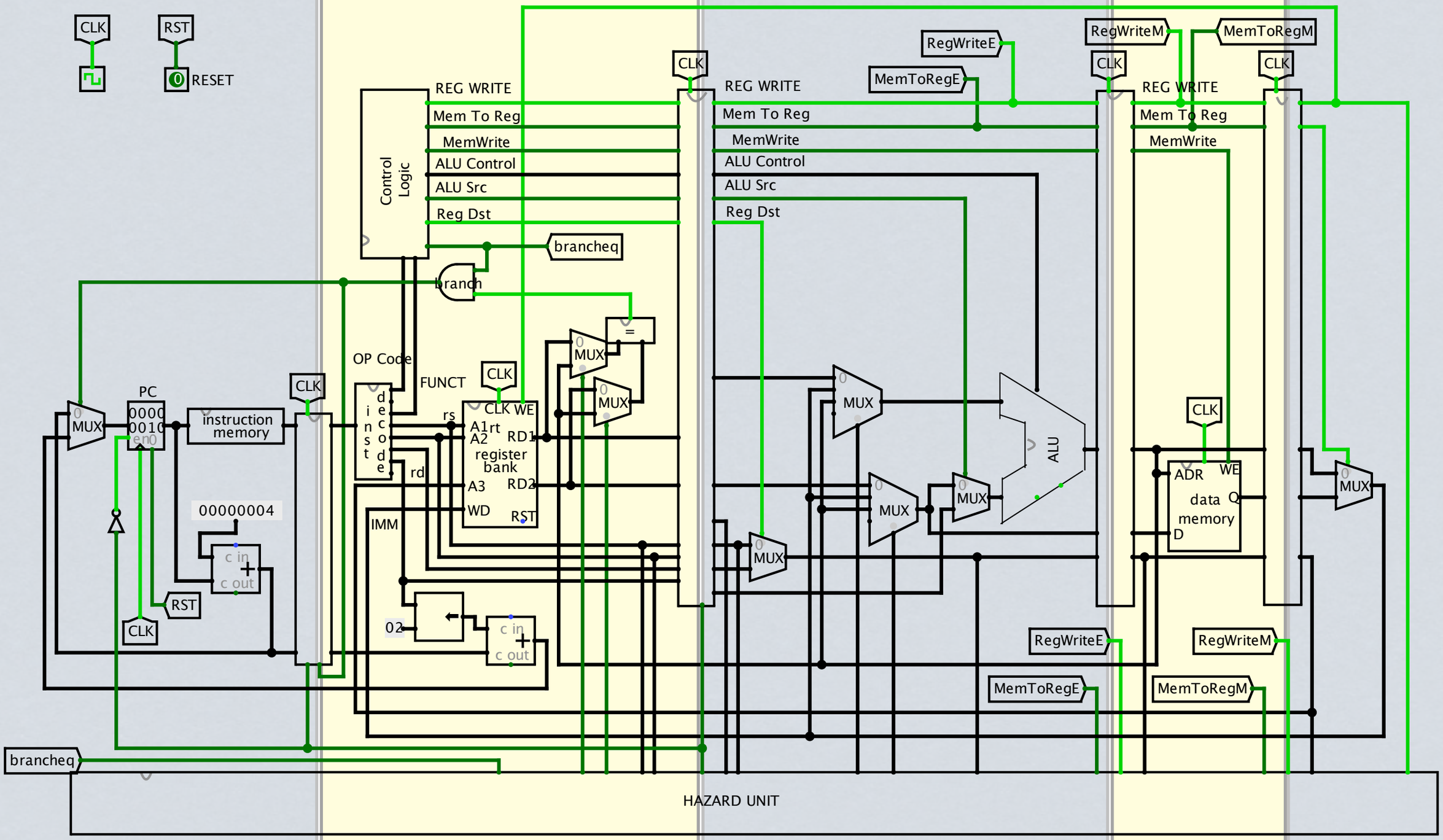
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



stall = rs(D) == rt(E) || rt(D) == rt(E) && MemtoReg(E)

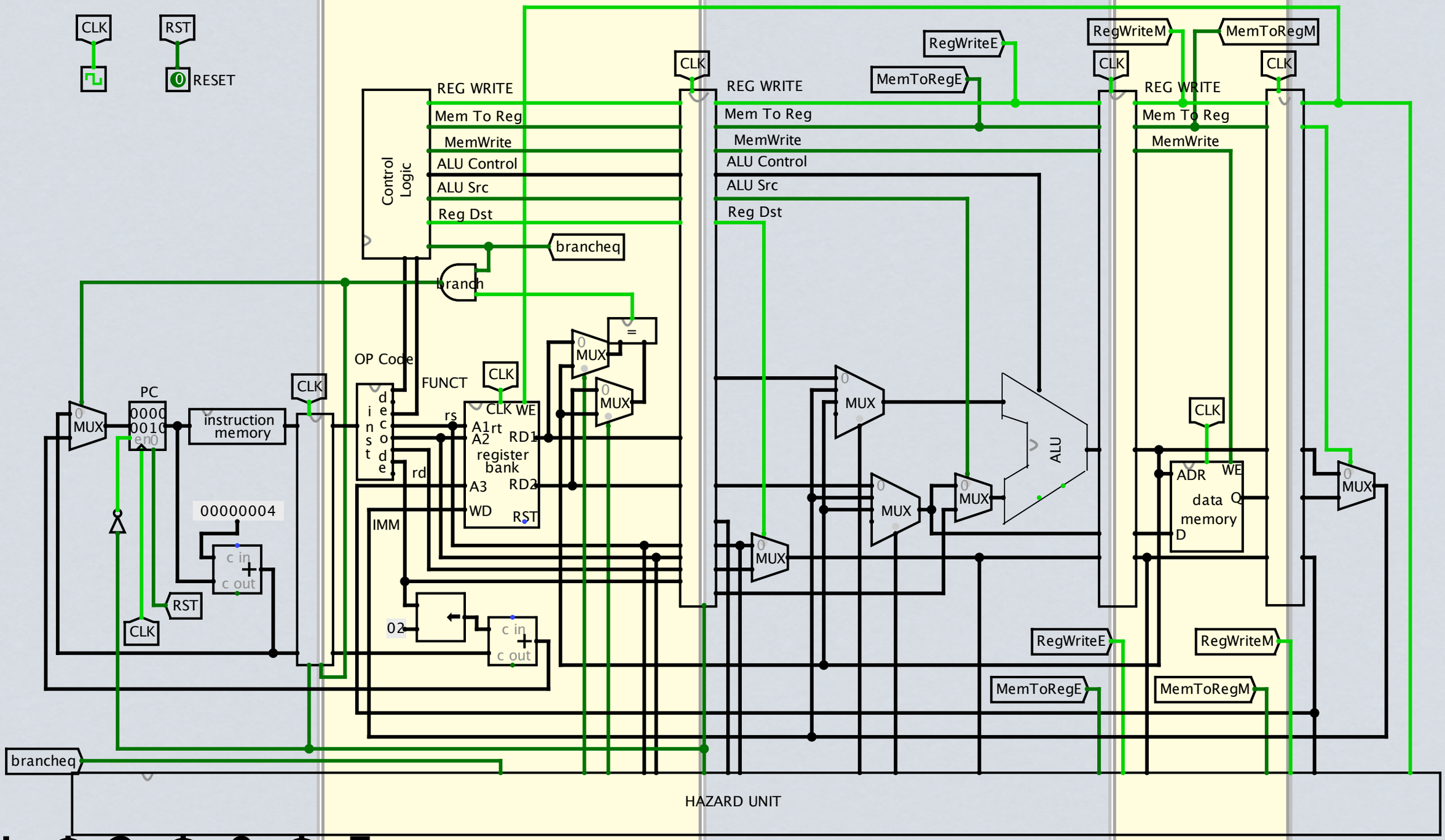
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



sub $t2, $s0, $s5

or $t1,$s4,$s0

and $t0, $s0, $s1

lw $s0, 40(

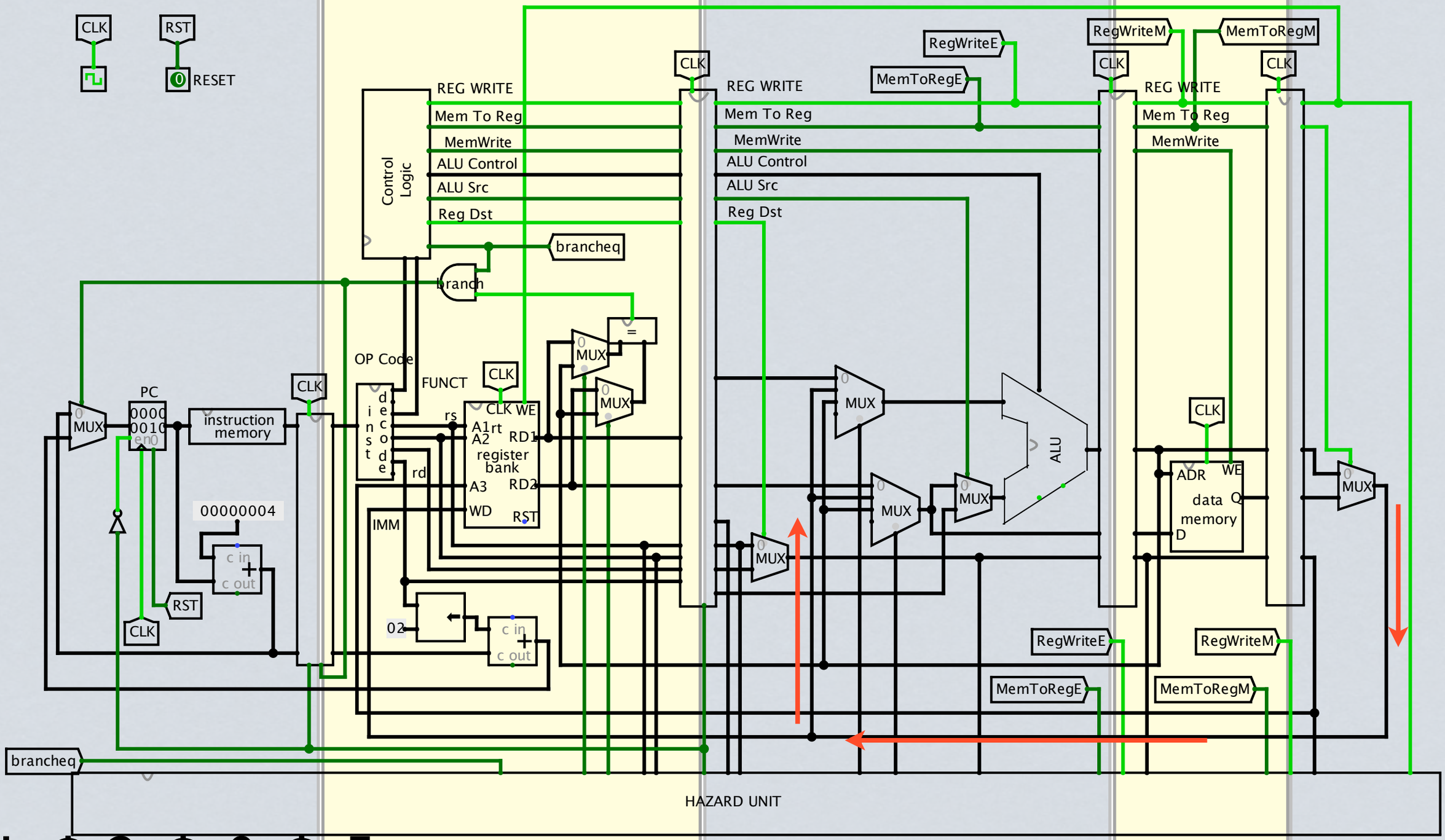
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



sub $t2, $s0, $s5

or $t1,$s4,$s0

and $t0, $s0, $s1

lw $s0, 40(

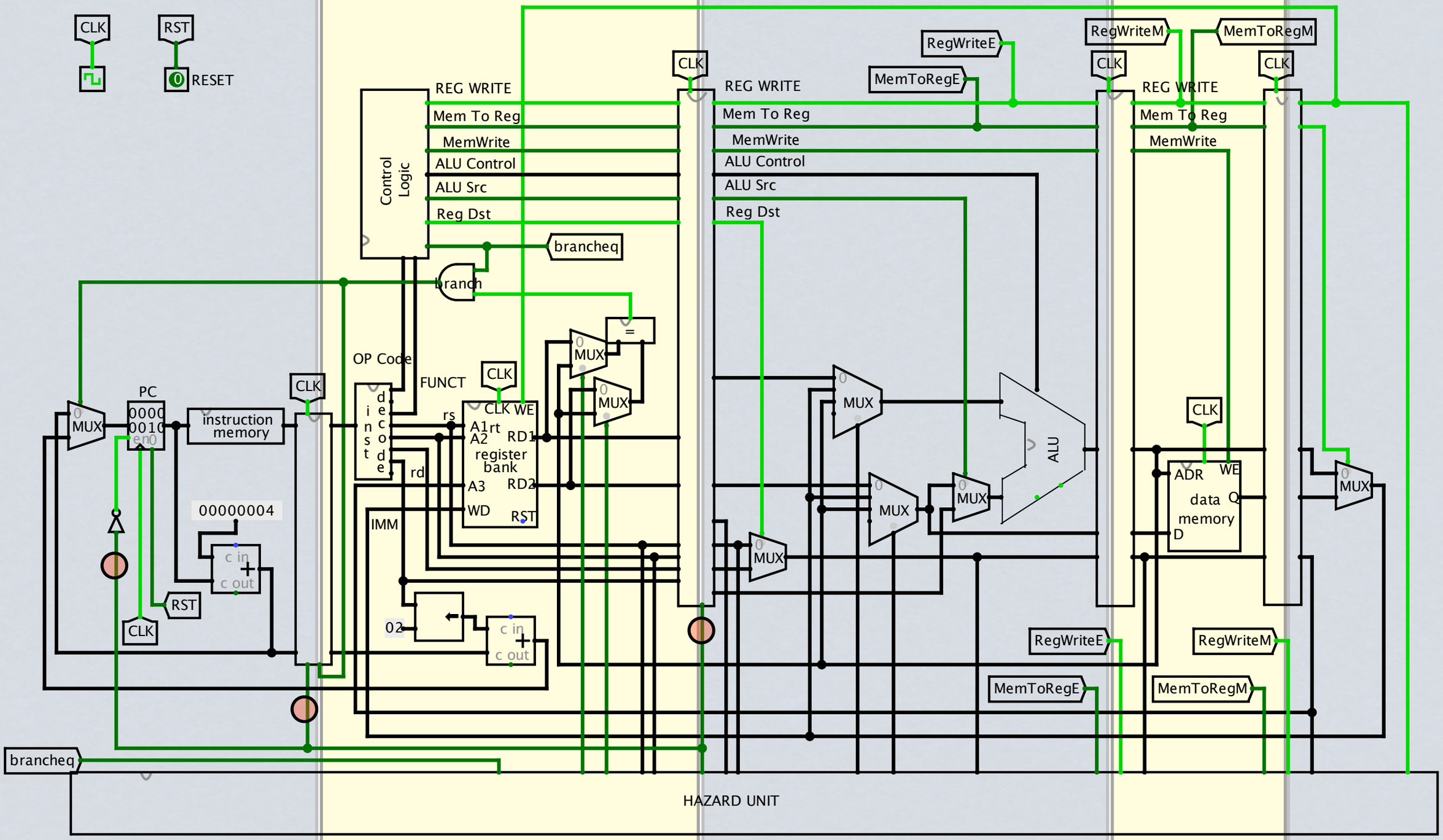
STAGE F

STAGE D

STAGE E

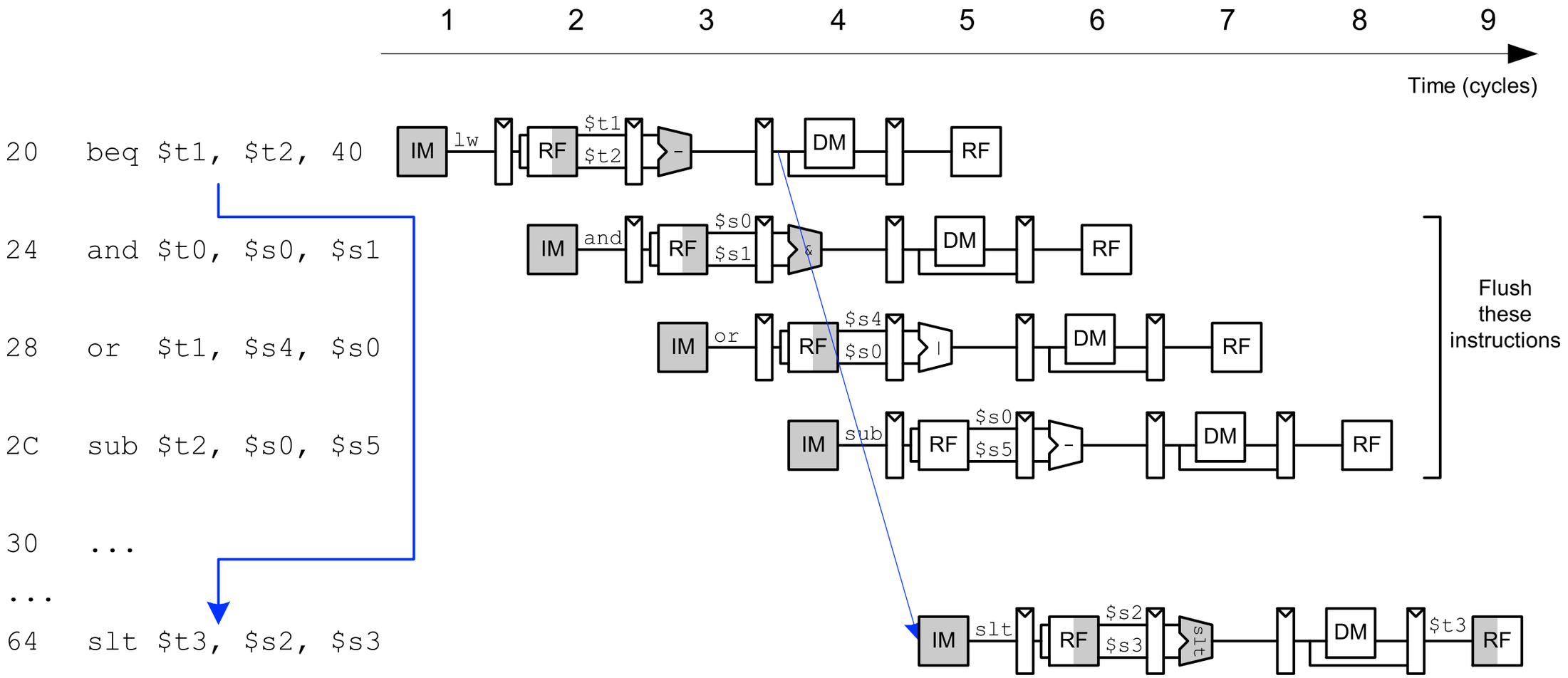
STAGE M

STAGE W

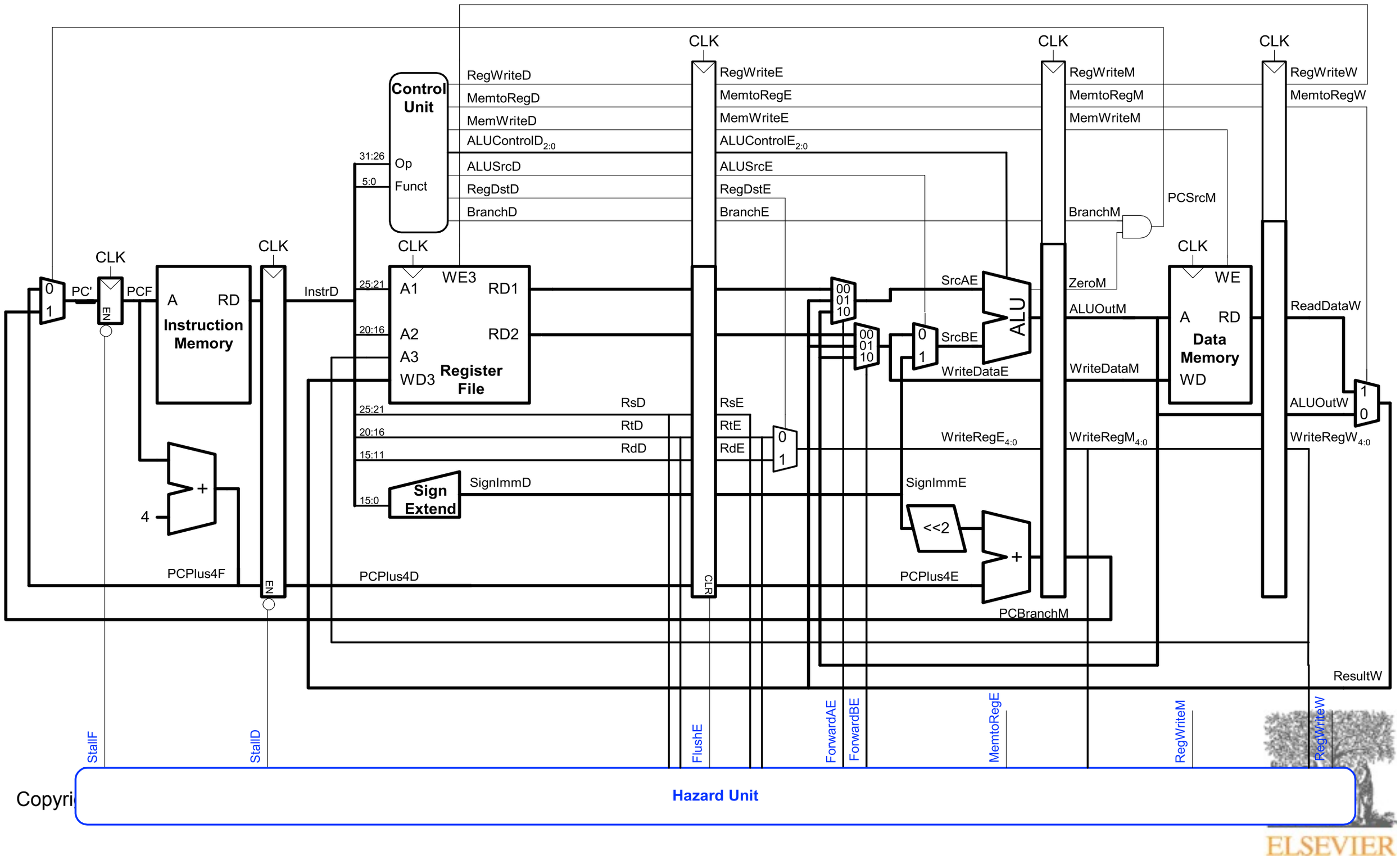


stall = rs(D) == rt(E) || rt(D) == rt(E) && MemtoReg(E)

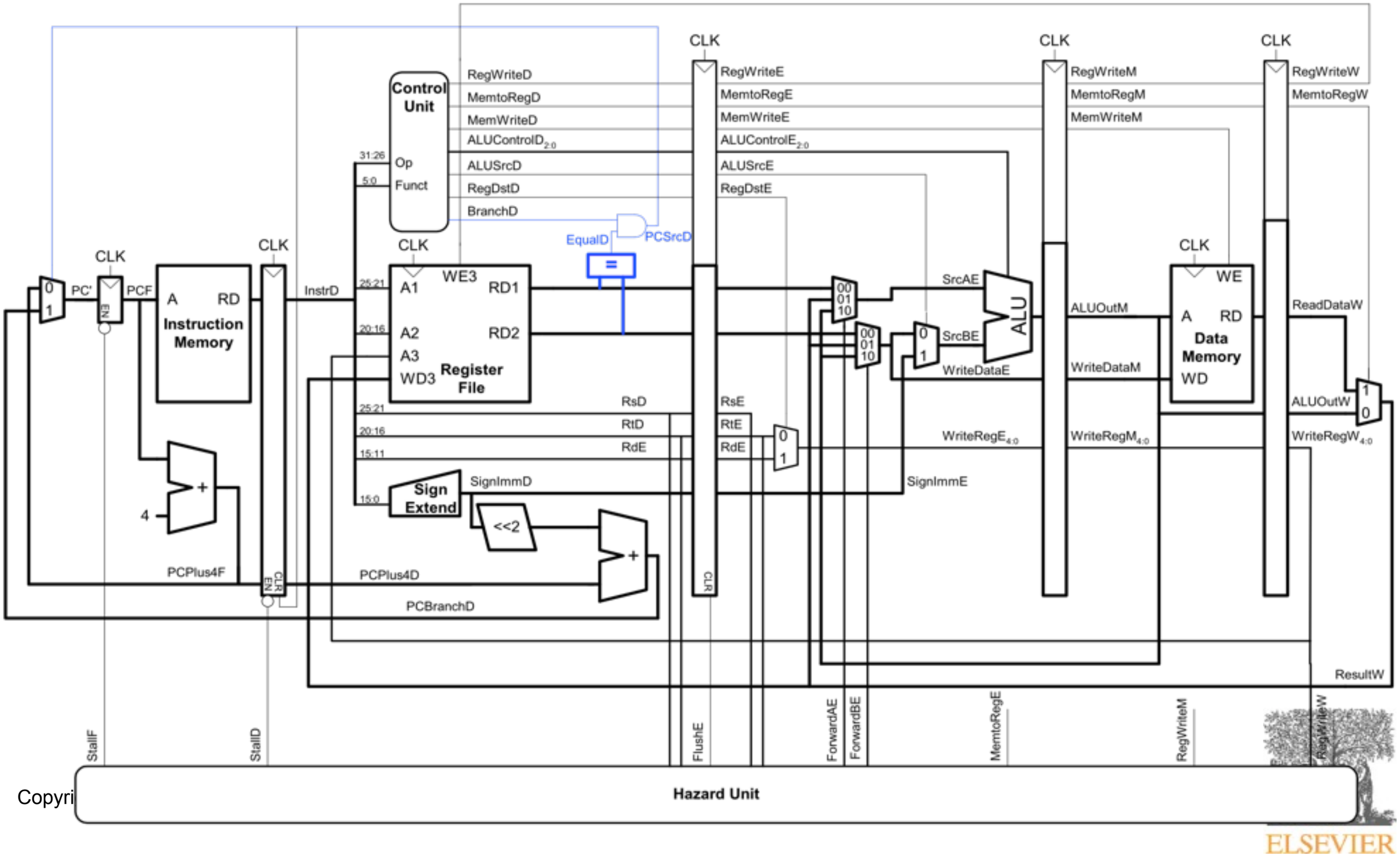
Control Hazards



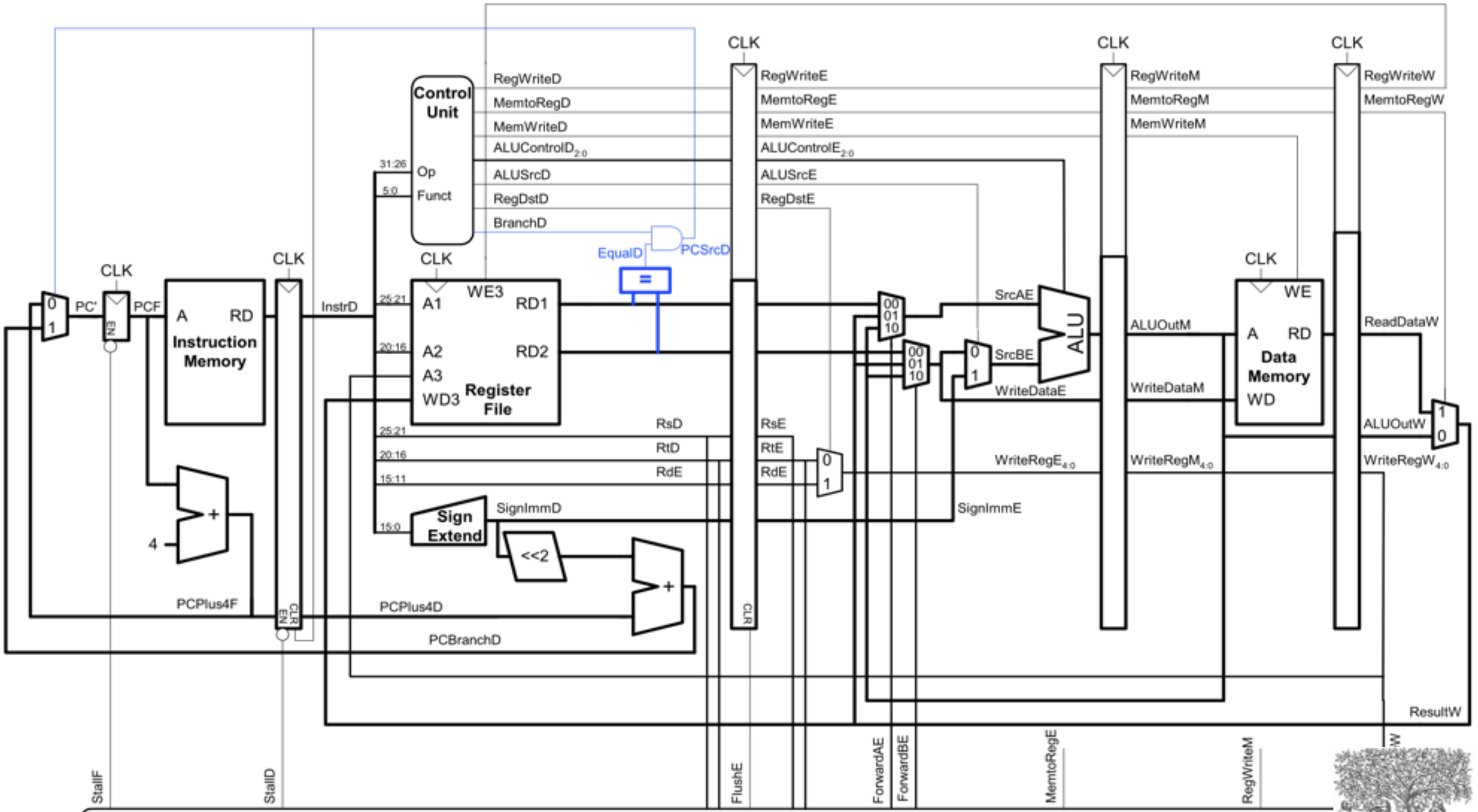
Control Hazards: Original Pipeline



Control Hazards: Early Branch Resolution

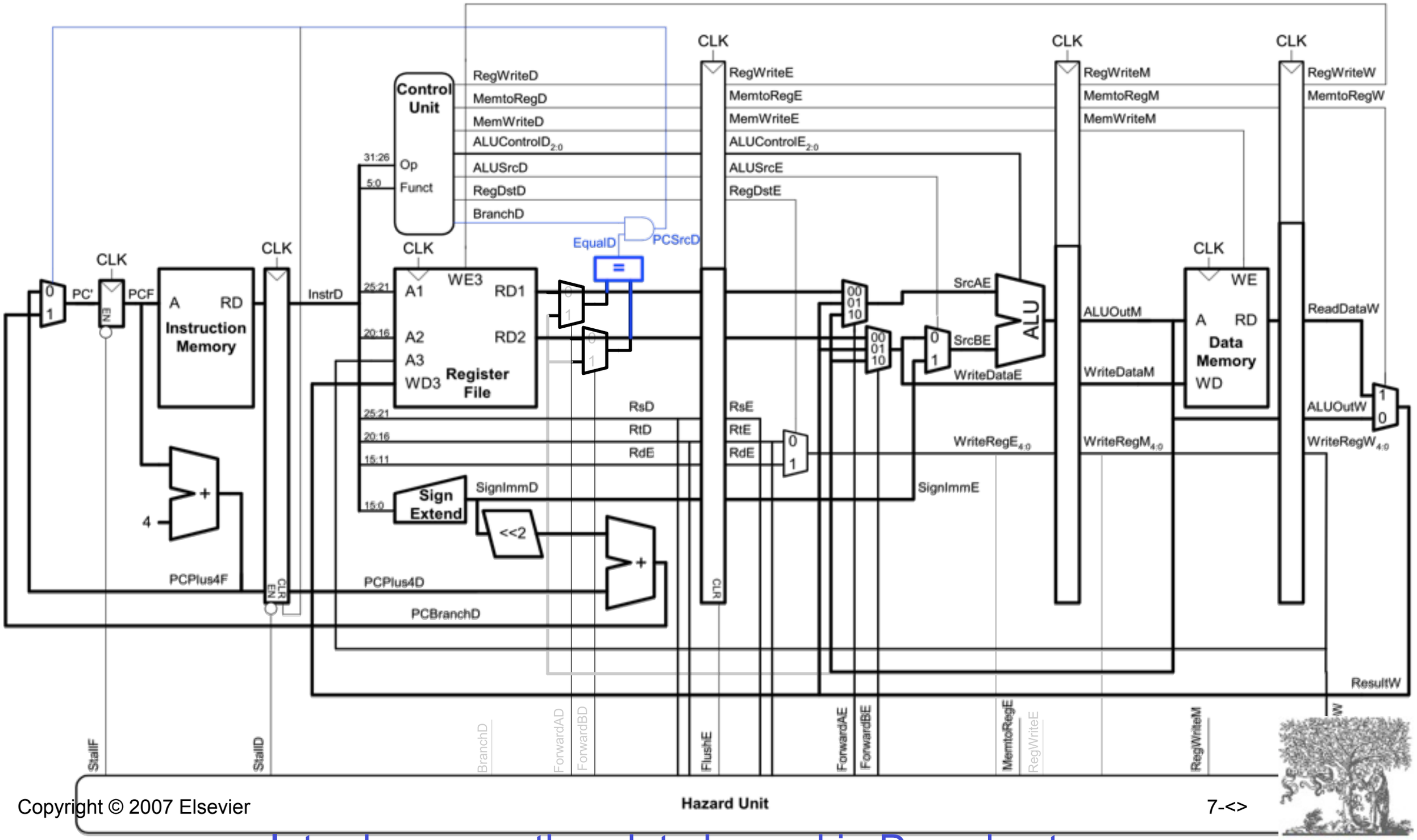


Control Hazards: Early Branch Resolution



Introduces another data hazard in Decode stage

Control Hazards: Early Branch Resolution



Introduces another data hazard in Decode stage

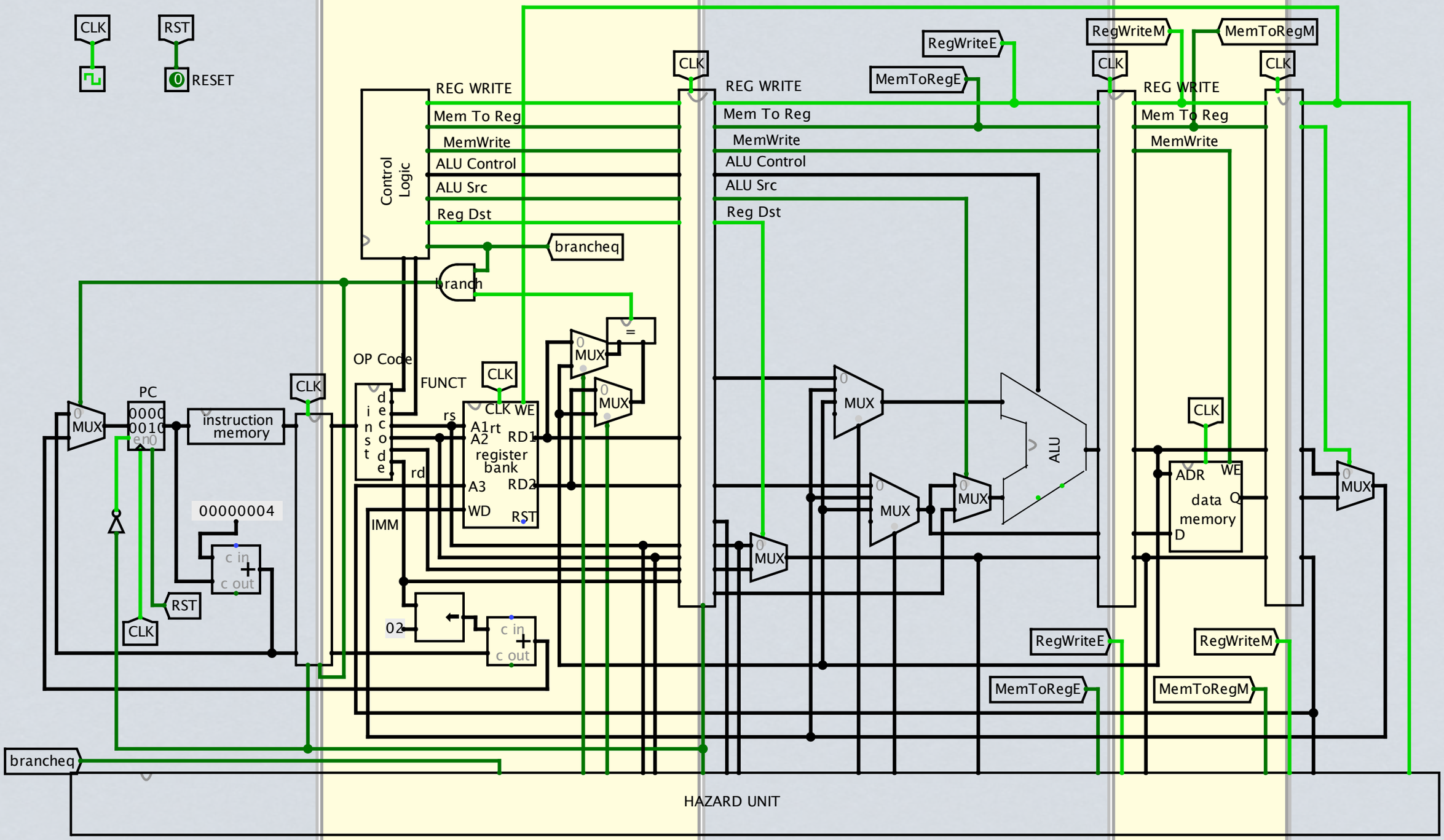
STAGE F

STAGE D

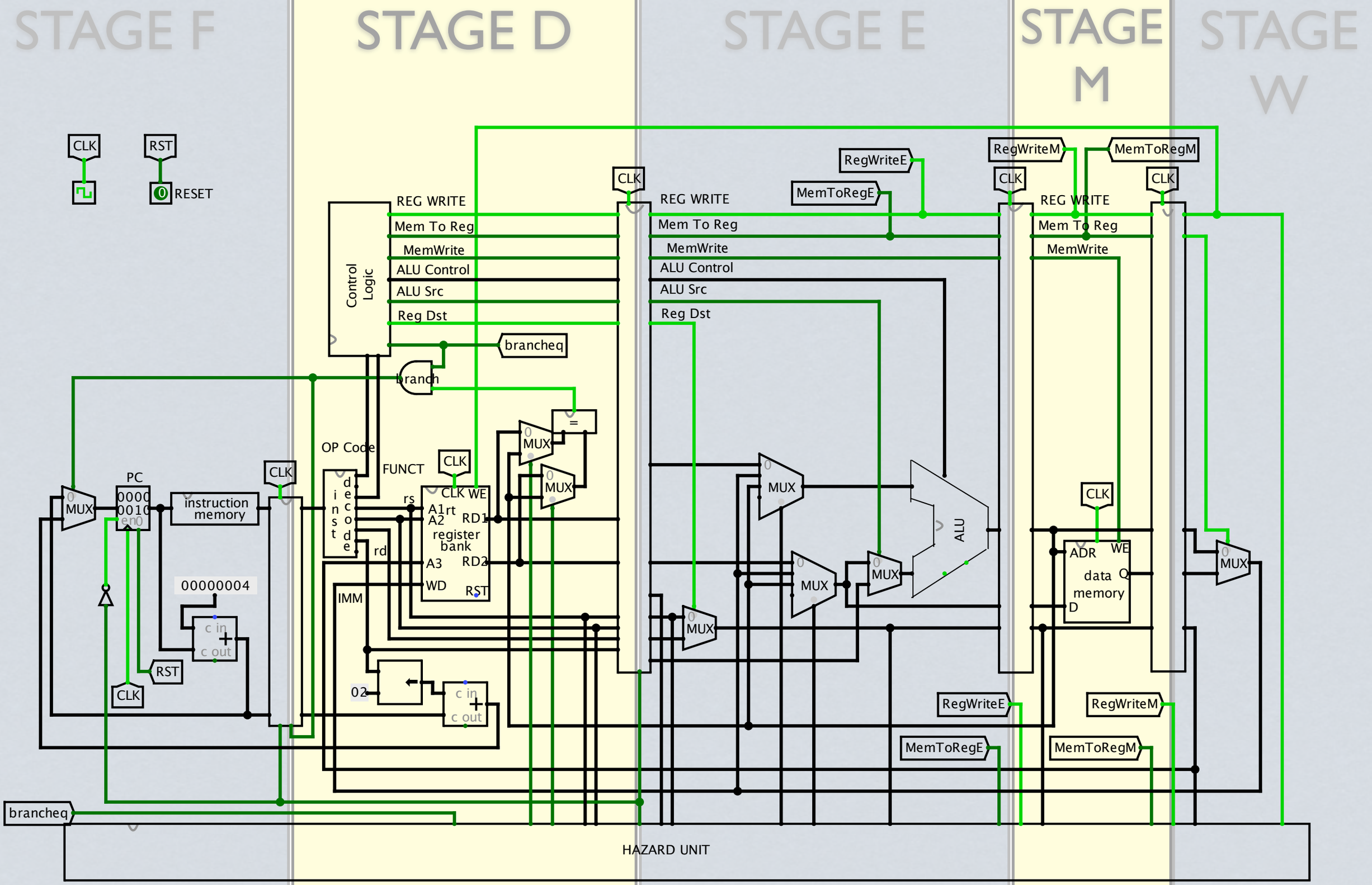
STAGE E

STAGE M

STAGE W



20 beq$t1,$t2,40



24 and$t0,$s0,$s1

20 beq$t1,$t2,40

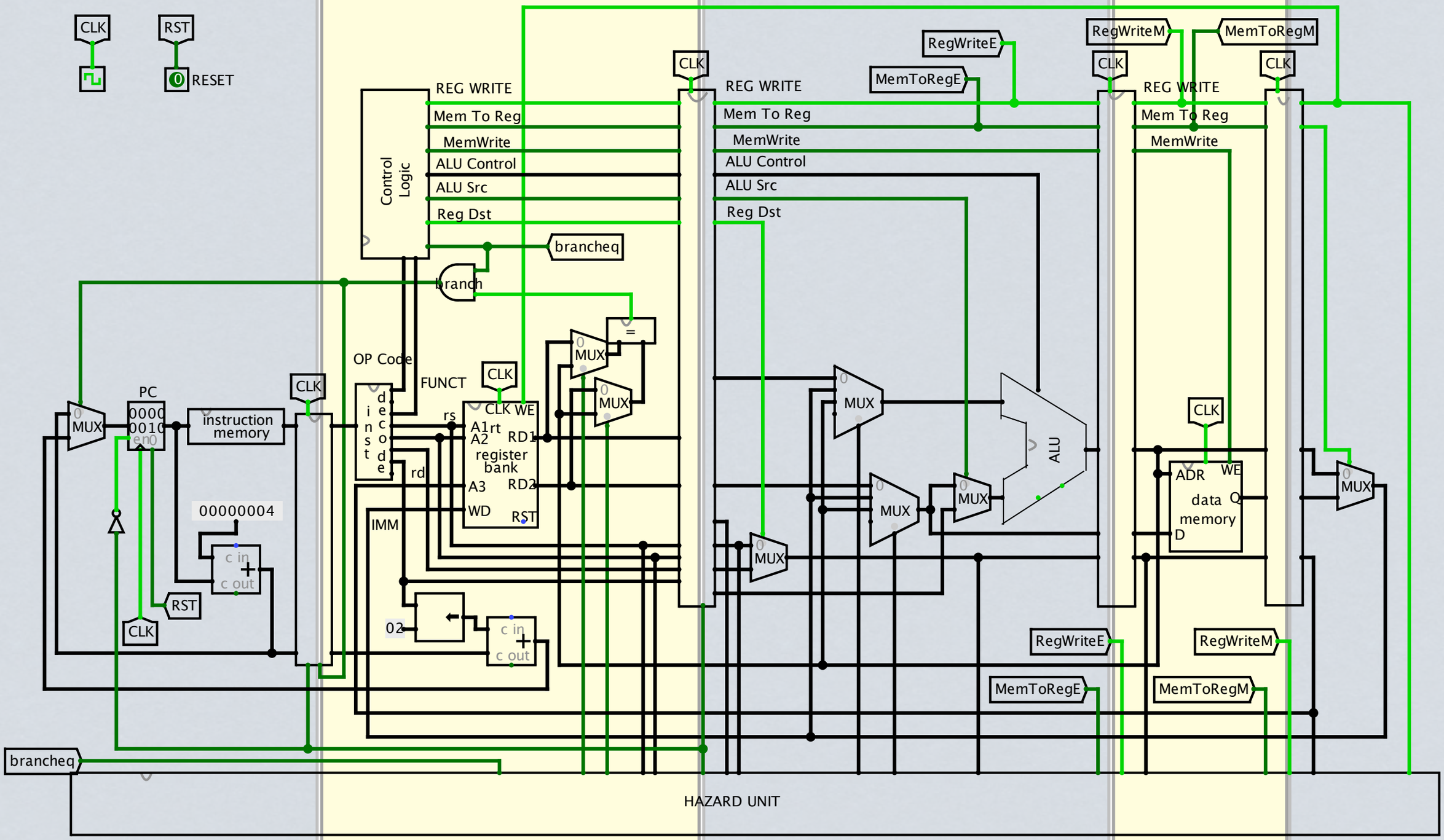
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



4 slt $t3, $s2, $s3

24 and$t0,$s0,$s1

20 beq$t1,$t2,40

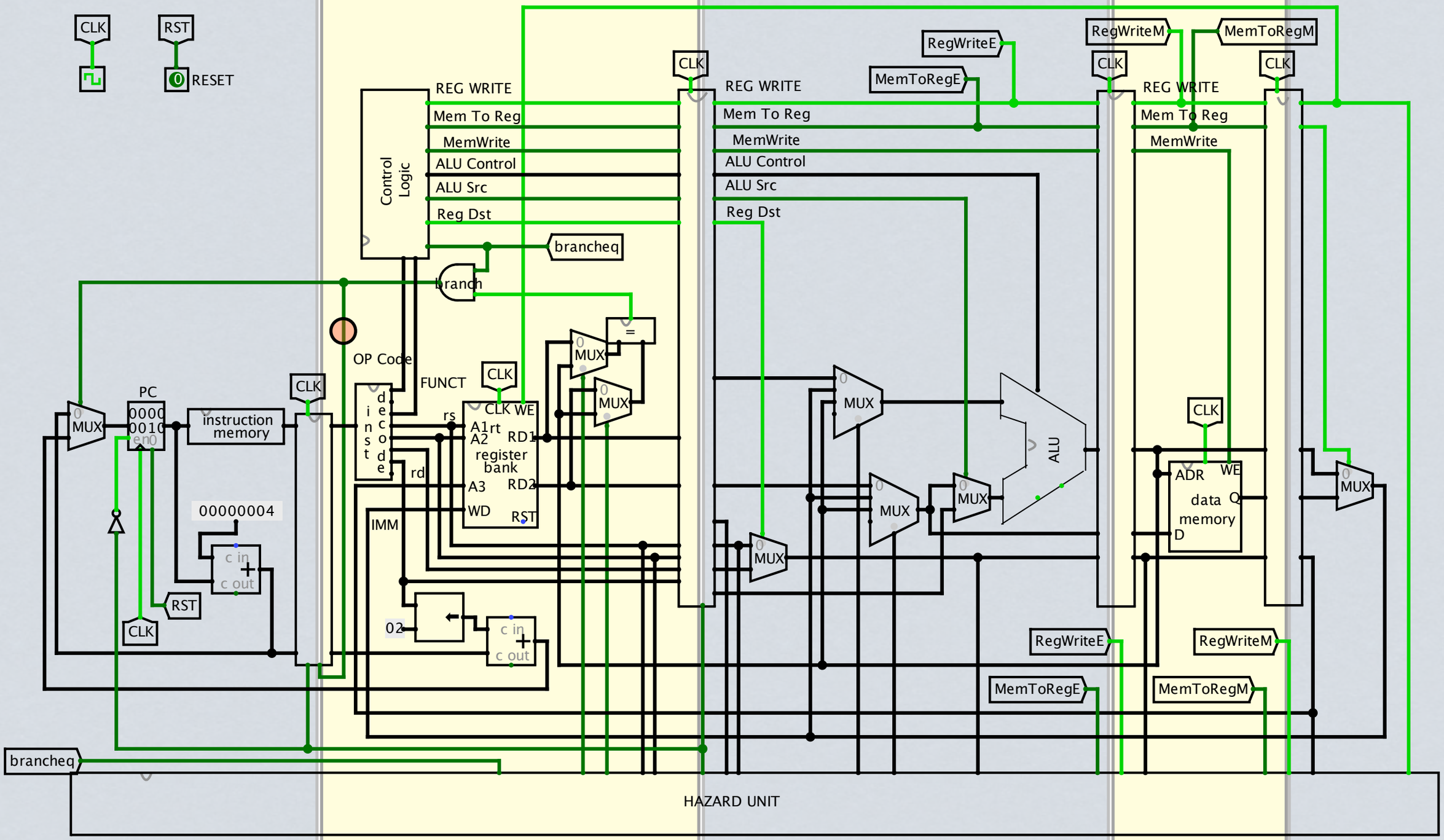
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



4 slt $t3, $s2, $s3

20 beq$t1,$t2,40

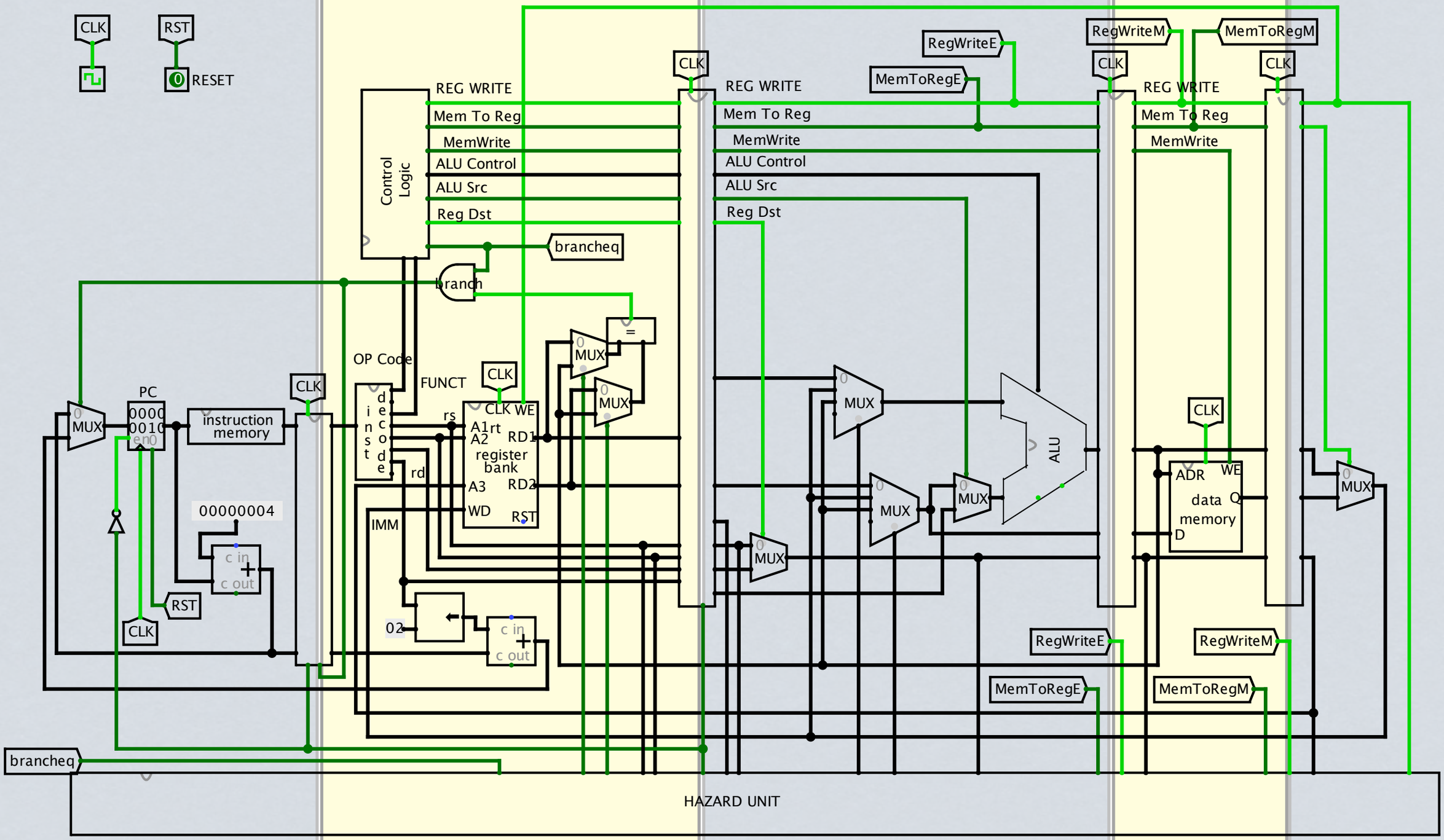
STAGE F

STAGE D

STAGE E

STAGE M

STAGE W



68 ...

64 slt $t3, $s2, $s3

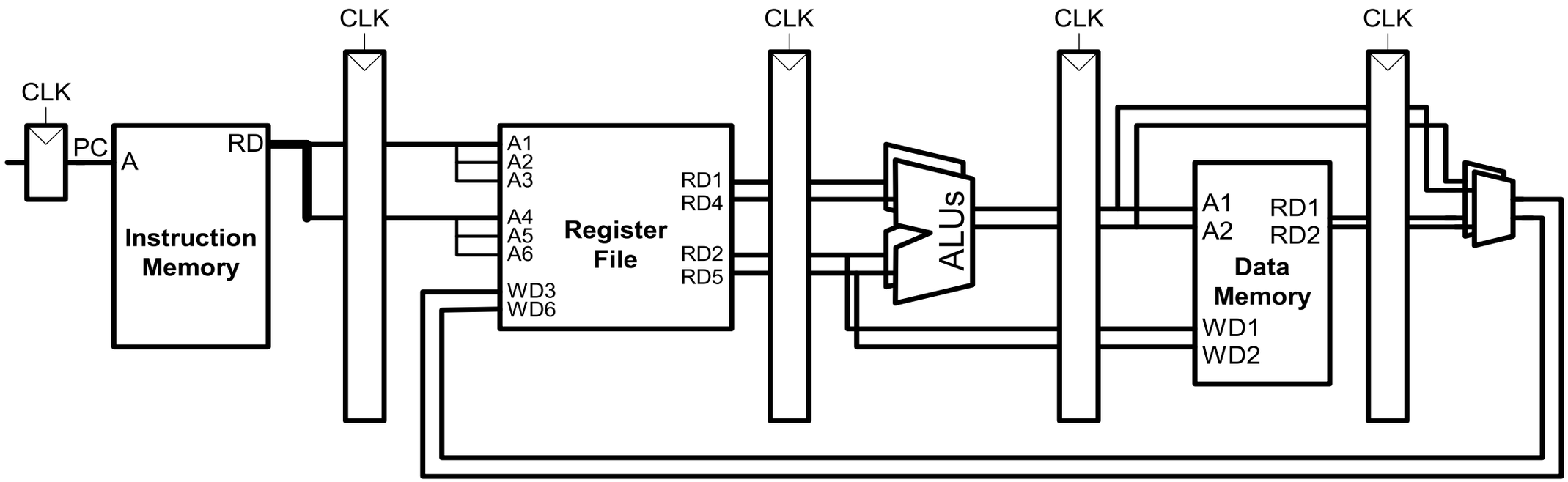
20 beq$t1,$t2,40

Other Techniques

- Delayed Branches
- Branch Prediction
- Compiler Optimization
- Vector/SIMD Machines
- Superscalar Organization

Superscalar

- Multiple copies of datapath execute multiple instructions at once
- Dependencies make it tricky to issue multiple instructions at once

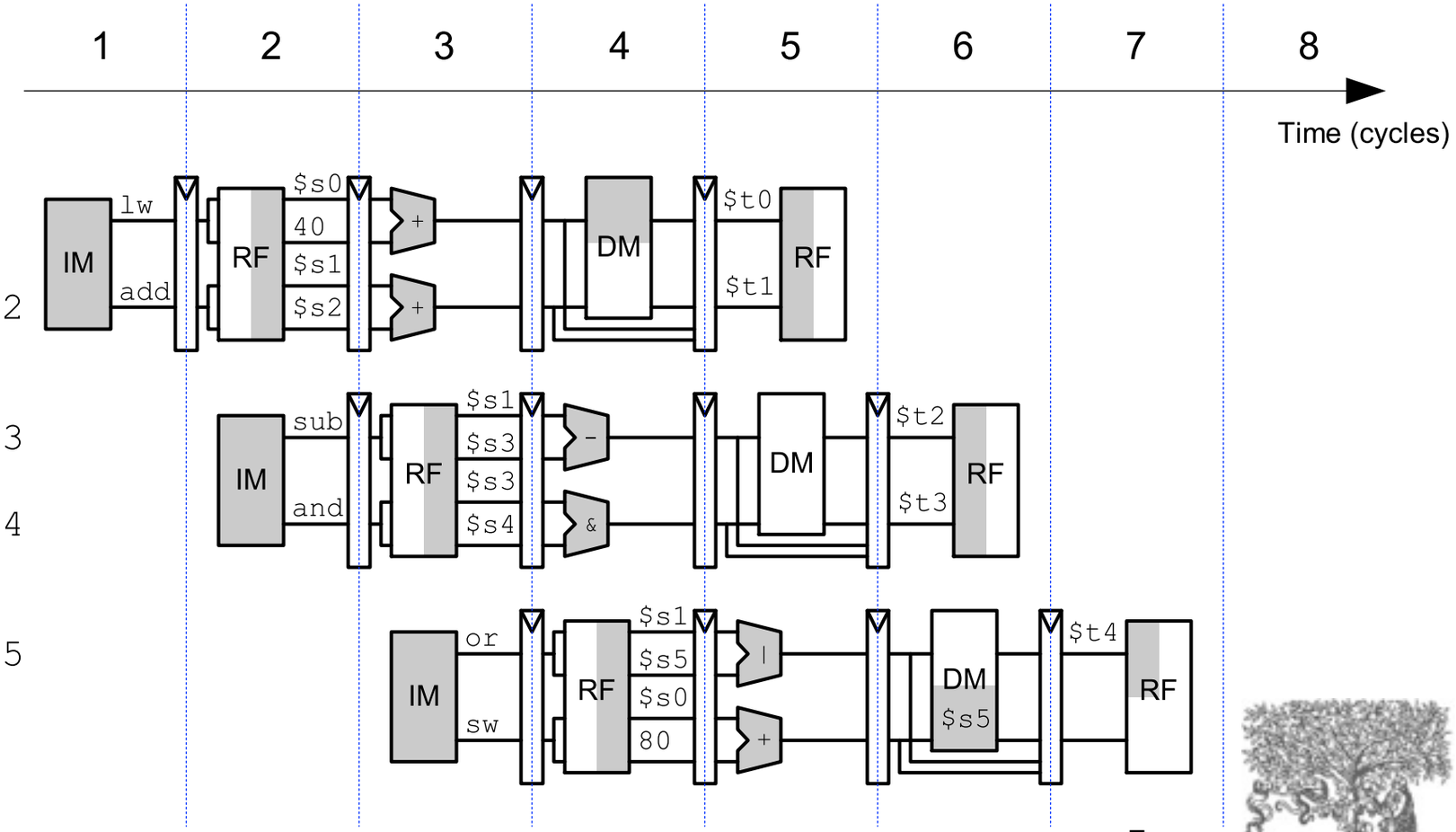


Superscalar Example

```
lw   $t0, 40($s0)
add  $t1, $t0, $s1
sub  $t0, $s2, $s3
and  $t2, $s4, $t0
or   $t3, $s5, $s6
sw   $s7, 80($t3)
```

Ideal IPC: 2

Actual IPC: 2



Limits

$$\lim_{n \to \infty} \sum_{x=1}^{n} 1/x = ?$$

Limits

```
int main( int argc, char * argv[] ) {

  float total = 0, x;
  int low = 1;
  int high = 10000000;

  for ( x = low; x <= high; x++ ) {
    total = total + 1/x;
  }

  printf( "total = %g\n", total );
}
```

Limits

```
int main( int argc, char * argv[] ) {

  float total = 0, x;
  int low = 1;
  int high = 10000000;

  for ( x = high; x >= low; x-- ) {
    total = total + 1/x;
  }

  printf( "total = %g\n", total );
}
```

Associating

$$(A + (B + C)) = ((A + B) + C)$$

Scientific Notation

6.022×10^{23} mol−1

2.9979 x 10^{8} m/s

mantissa exponent