# Timing analysis of a task running in isolation

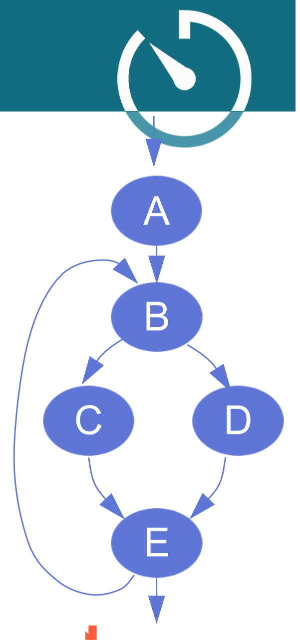
# What do we want to compute?

```
even = 0;
odd = 0;
for (i=0 ; i<255 ; i++) {
  if (i%2 == 0)
    even++;
  else
    odd++;
```

compiler

1100101

CFG builder




flow analysis

timing analysis

$$\max\ T = x_A.t_A + x_B.t_B + x_C.t_C + x_D.t_D + x_E.t_E$$

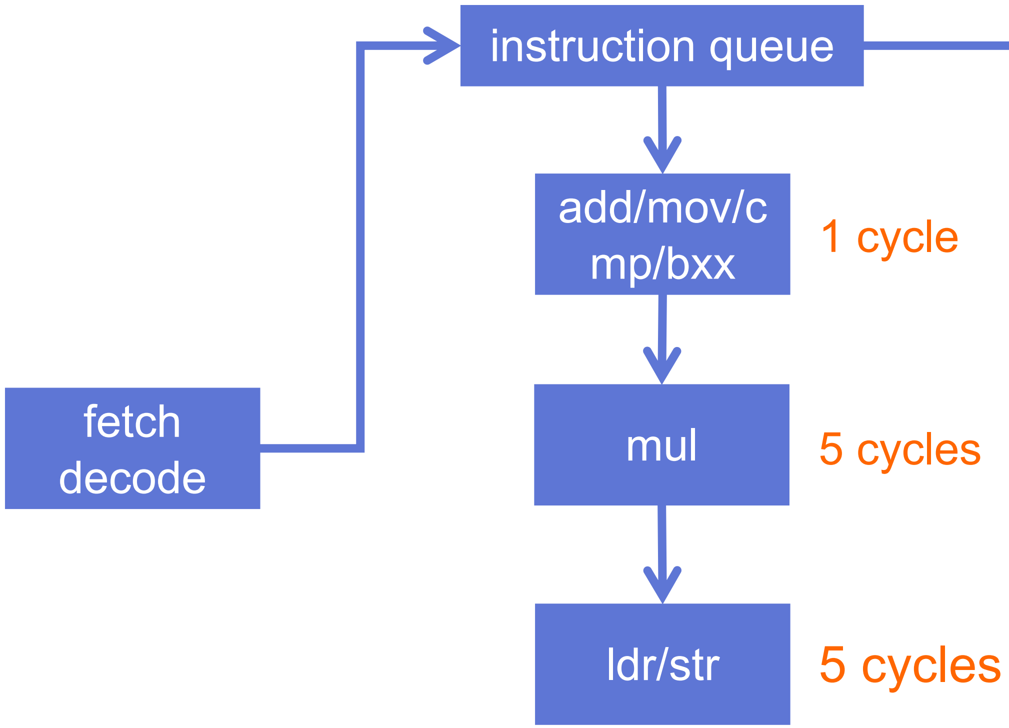
| | |
|---|---|
| $x_A = 1$ | $x_A = x_{AB}$ |
| $x_B = x_{AB} + x_{EB}$ | $x_B = x_{BC} + x_{BD}$ |
| $x_C = x_{BC}$ | $x_C = x_{CE}$ |
| $x_D = x_{BD}$ | $x_D = x_{DE}$ |
| $x_E = x_{CE} + x_{DE}$ | $x_E = 1 + x_{EB}$ |

$t_A = \dots;\ t_B = \dots;\ t_C = \dots;\ t_D = \dots;\ t_E = \dots;$

$x_{EB} \leq 255;\qquad x_{BC} \leq 0.5\ xB;$

# Pipelined execution

# Running example: execution pipeline




one instruction fetched each cycle (in the program order)

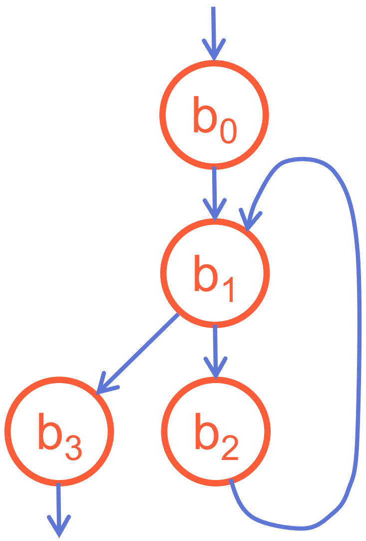
one instruction issued each cycle (in the program order)

one instruction committed each cycle (in the program order)

branches are predicted "not taken"

# Running example: task code

```c
int product = 1;
int init_val;
for (int i=0; i<N ; i++){
    product *= i;
    a[i] = init_val;
}
```



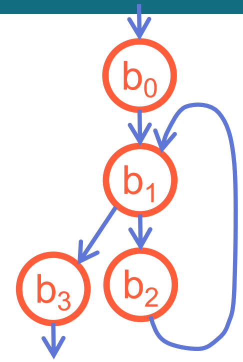

```
__start:    mov r0,#1      @ product
            mov r1,#0      @ i
            adr r2,a
            adr r3,init_val
            ldr r3,[r3]
```
$b_0$

```
while:      mul r0,r1,r0
            cmp r1,#N
            bhs end
```
$b_1$

```
            str r3,[r2],#4
            add r1,r1,#1
            b while
```
$b_2$

```
end:        adr r2,product
            str r0,[r2]
```
$b_3$

# Early approaches




```
0a   __start:  mov r0,#1
0b             mov r1,#0
0c             adr r2,a
0d             adr r3,init_val
0e             ldr r3,[r3]
1a   while:    mul r0,r1,r0
1b             cmp r1,#N
1c             bhs end
2a             str r3,[r2],#4
2b             add r1,r1,#1
2c             b while
3a   end:      adr r2,product
3b             str r0,[r2]
```

$b_1$ after $b_0$

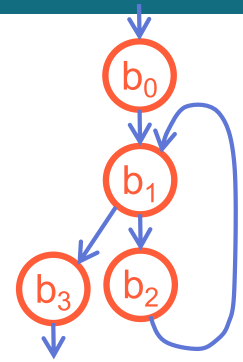
$t_{0\text{-}1}$ = 3 cycles

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **F** | 0a | 0b | 0c | 0d | 0e | 1a | 1b | 1c | | | | | | | | | | |
| **A** | | 0a | 0b | 0c | 0d | | | 1b | 1c | | | | | | | | | |
| **Mu** | | | | | | | 1a | 1a | 1a | 1a | 1a | | | | | | | |
| **Me** | | | | | | 0e | 0e | 0e | 0e | 0e | | | | | | | | |
| **C** | | | 0a | 0b | 0c | 0d | | | | | 0e | 1a | 1b | 1c | | | | |

$b_1$ after $b_2$

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **F** | | | | | | | | | | | | | | | | | | |
| **A** | | | | | | | | | | | | | | | | | | |
| **Mu** | | | | | | | | | | | | | | | | | | |
| **Me** | | | | | | | | | | | | | | | | | | |
| **C** | | | | | | | | | | | | | | | | | | |

# Early approaches




```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```

$b_1$ after $b_0$ $t_{0\text{-}1}$ = 3 cycles

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **F** | 0a | 0b | 0c | 0d | 0e | 1a | 1b | 1c | | | | | | | | | | | |
| **A** | | 0a | 0b | 0c | 0d | | | 1b | 1c | | | | | | | | | | |
| **Mu** | | | | | | | 1a | 1a | 1a | 1a | 1a | | | | | | | | |
| **Me** | | | | | | 0e | 0e | 0e | 0e | 0e | | | | | | | | | |
| **C** | | | 0a | 0b | 0c | 0d | | | | | 0e | 1a | 1b | 1c | | | | | |

$b_1$ after $b_2$ $t_{2\text{-}1}$ = 4 cycles

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **F** | 2a | 2b | 2c | x | 1a | 1b | 1c | | | | | | | | | | | | |
| **A** | | | 2b | 2c | | | 1b | 1c | | | | | | | | | | | |
| **Mu** | | | | | | 1a | 1a | 1a | 1a | 1a | | | | | | | | | |
| **Me** | | 2a | 2a | 2a | 2a | 2a | | | | | | | | | | | | | |
| **C** | | | | | | | 2a | 2b | 2c | | 1a | 1b | 1c | | | | | | |

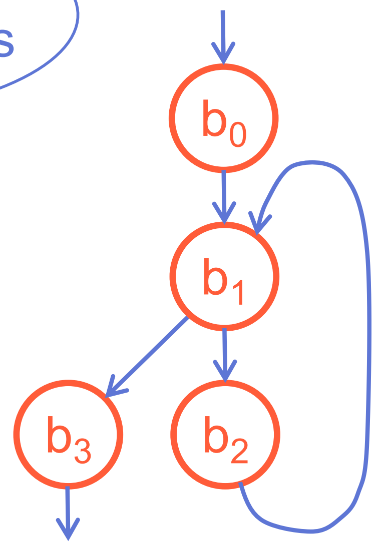
# Improved ILP formulation (IPET)

- max $T = \Sigma\, x_{i\text{-}j} \,.\, t_{i\text{-}j}$

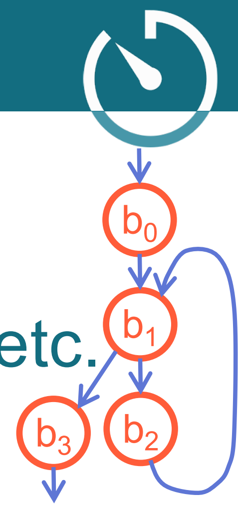
computed using reservation tables

- with:
  - $x_0 = x_{0\text{-}1} = 1$
  - $x_1 = x_{0\text{-}1} + x_{2\text{-}1}$
  - $x_1 = x_{1\text{-}3} + x_{1\text{-}2}$
  - $x_2 = x_{1\text{-}2}$
  - $x_2 = x_{2\text{-}1}$

  - $x_{2\text{-}1} \leq N \,.\, x_{0\text{-}1}$



# Early approaches




## Limitations on reservation tables

- complex pipelines: superscalar, ooo, multiple FUs, etc.
- initial state/history: long timing effects

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```

$b_1$ after $b_0$ — $t_0$ = 11 cycles $t_{0\text{-}1}$ = 3 cycles

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **F** | 0a | 0b | 0c | 0d | 0e | 1a | 1b | 1c | | | | | | | | | | |
| **A** | | 0a | 0b | 0c | 0d | | | 1b | 1c | | | | | | | | | |
| **Mu** | | | | | | | 1a | 1a | 1a | 1a | 1a | | | | | | | |
| **Me** | | | | | | 0e | 0e | 0e | 0e | 0e | | | | | | | | |
| **C** | | | 0a | 0b | 0c | 0d | | | | | 0e | 1a | 1b | 1c | | | | |

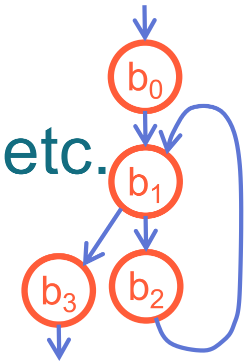
$b_2$ after $b_1$ — $t_{1\text{-}2}$ = 3 cycles

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **F** | 1a | 1b | 1c | 2a | 2b | 2c | | | | | | | | | | | | |
| **A** | | | 1b | 1c | | 2b | 2c | | | | | | | | | | | |
| **Mu** | | 1a | 1a | 1a | 1a | 1a | | | | | | | | | | | | |
| **Me** | | | | | 2a | 2a | 2a | 2a | 2a | | | | | | | | | |
| **C** | | | | | | | 1a | 1b | 1c | 2a | 2b | 2c | | | | | | |

# Early approaches

## Limitations on reservation tables

- complex pipelines: superscalar, ooo, multiple FUs, etc.
- initial state/history: long timing effects




```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```

$t_0$ = 11 cycles   $t_{0\text{-}1}$ = 3 cycles   $t_{1\text{-}2}$ = 3 cycles

$t_{0\text{-}1\text{-}2} = t_0 + t_{0\text{-}1} + t_{1\text{-}2}$ = 17 cycles

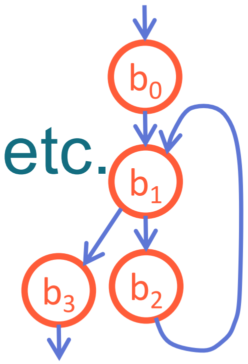
$b_2$ after $b_1$ after $b_0$

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **F** | | | | | | | | | | | | | | | | | | |
| **A** | | | | | | | | | | | | | | | | | | |
| **Mu** | | | | | | | | | | | | | | | | | | |
| **Me** | | | | | | | | | | | | | | | | | | |
| **C** | | | | | | | | | | | | | | | | | | |

# Early approaches

## Limitations on reservation tables

- complex pipelines: superscalar, ooo, multiple FUs, etc.
- initial state/history: long timing effects




```
0a   __start:  mov r0,#1
0b             mov r1,#0
0c             adr r2,a
0d             adr r3,init_val
0e             ldr r3,[r3]
1a   while:    mul r0,r1,r0
1b             cmp r1,#N
1c             bhs end
2a             str r3,[r2],#4
2b             add r1,r1,#1
2c             b while
3a   end:      adr r2,product
3b             str r0,[r2]
```

$t_0$ = 11 cycles  $t_{0\text{-}1}$ = 3 cycles  $t_{1\text{-}2}$ = 3 cycles

$t_{0\text{-}1\text{-}2} = t_0 + t_{0\text{-}1} + t_{1\text{-}2}$ = 17 cycles

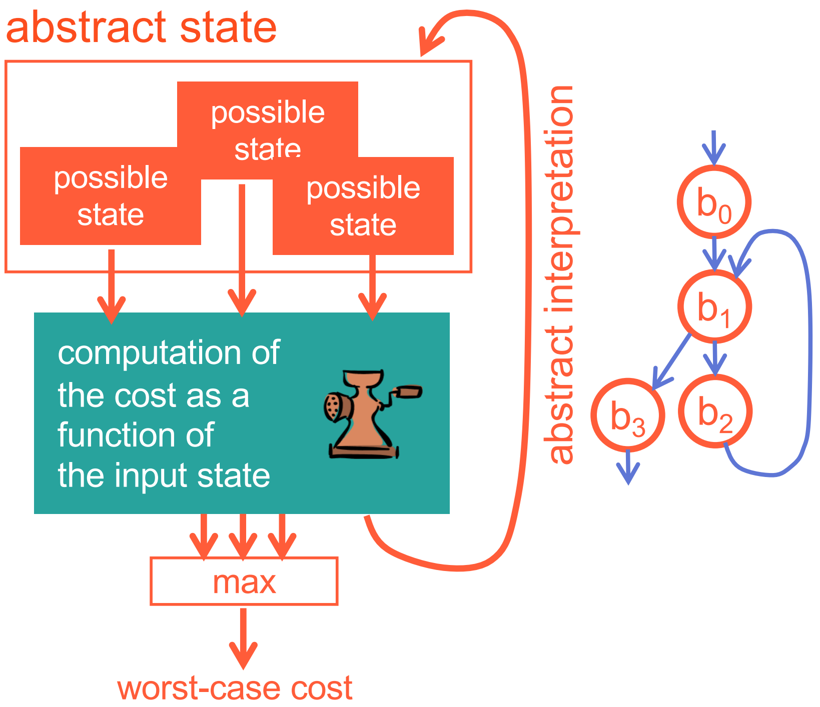
$b_2$ after $b_1$ after $b_0$

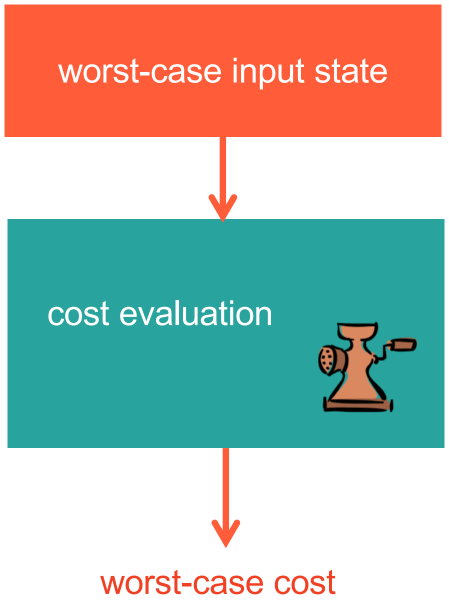
| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **F** | 0a | 0b | 0c | 0d | 0e | 1a | 1b | 1c | 2a | 2b | 2c | | | | | | | |
| **A** | | 0a | 0b | 0c | 0d | | | 1b | 1c | | 2b | 2c | | | | | | |
| **Mu** | | | | | | | 1a | 1a | 1a | 1a | 1a | | | | | | | |
| **Me** | | | | | | 0e | 0e | 0e | 0e | 0e | 2a | 2a | 2a | 2a | 2a | | | |
| **C** | | | 0a | 0b | 0c | 0d | | | | | 0e | 1a | 1b | 1c | | 2a | 2b | 2c |

$t_{0\text{-}1\text{-}2}$ = 18 cycles 😧

# Modern approaches

## Abstract interpretation vs.local analysis

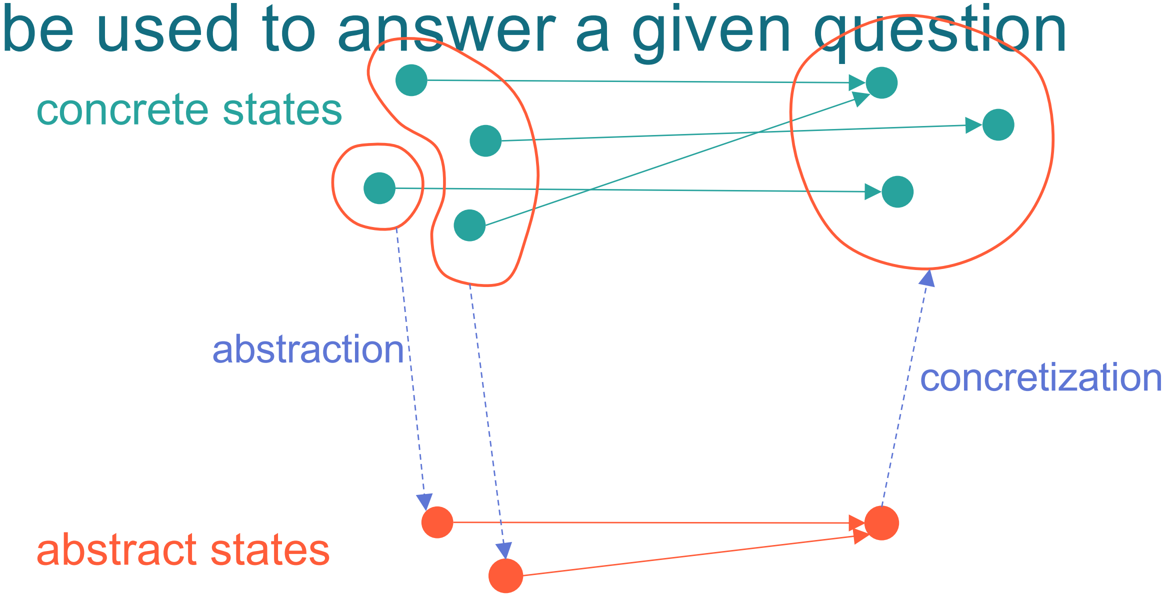


# What is Abstract Interpretation?

## A theory of approximation of the semantics of programs

- partial execution of a program to derive information that can be used to answer a given question

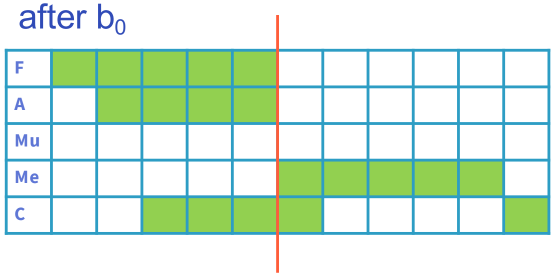


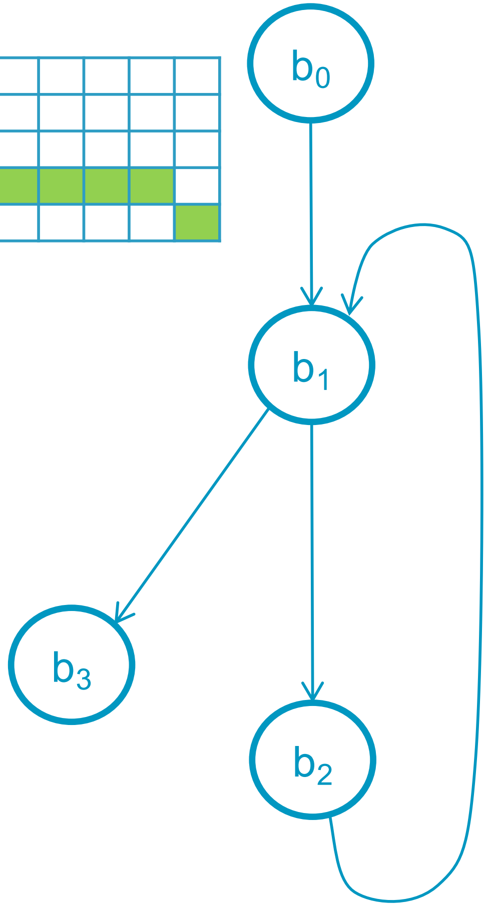

P. Cousot, R. Cousot, *Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints*, POPL, 1977

# A sketch of pipeline analysis by abstract interpretation

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```

after $b_0$



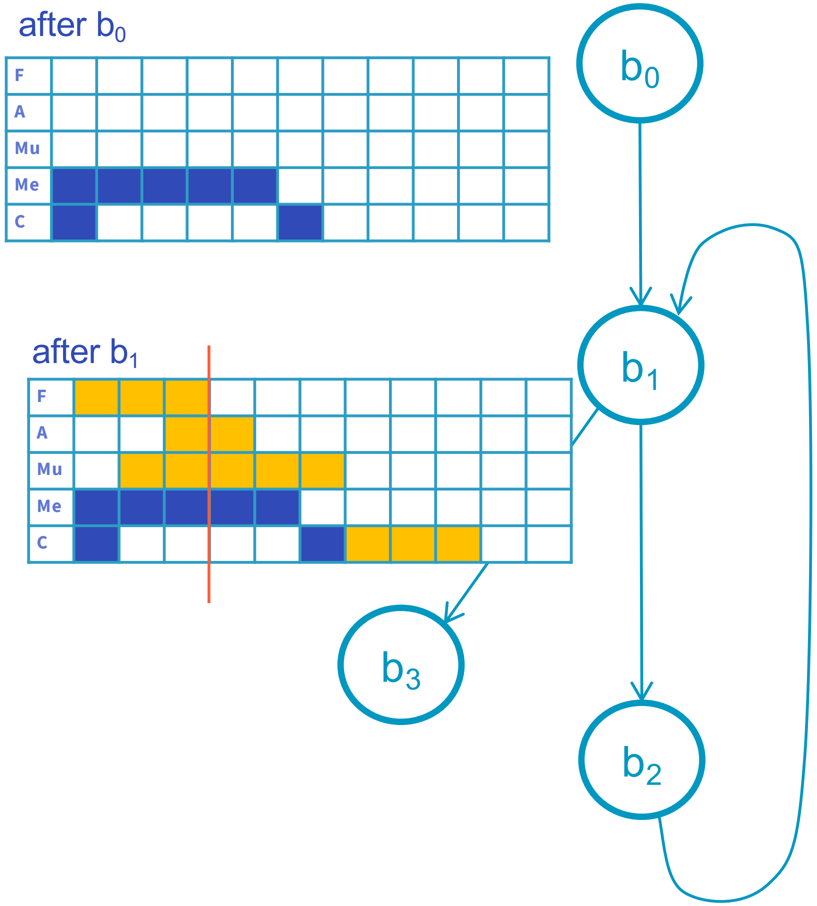




$t_0 = 11$

**Disclaimer**: this is only a possible way to analyze a pipeline with "abstract" interpretation.
This is not the way it is done in tools that use this technique.

# A sketch of pipeline analysis by abstract interpretation

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```

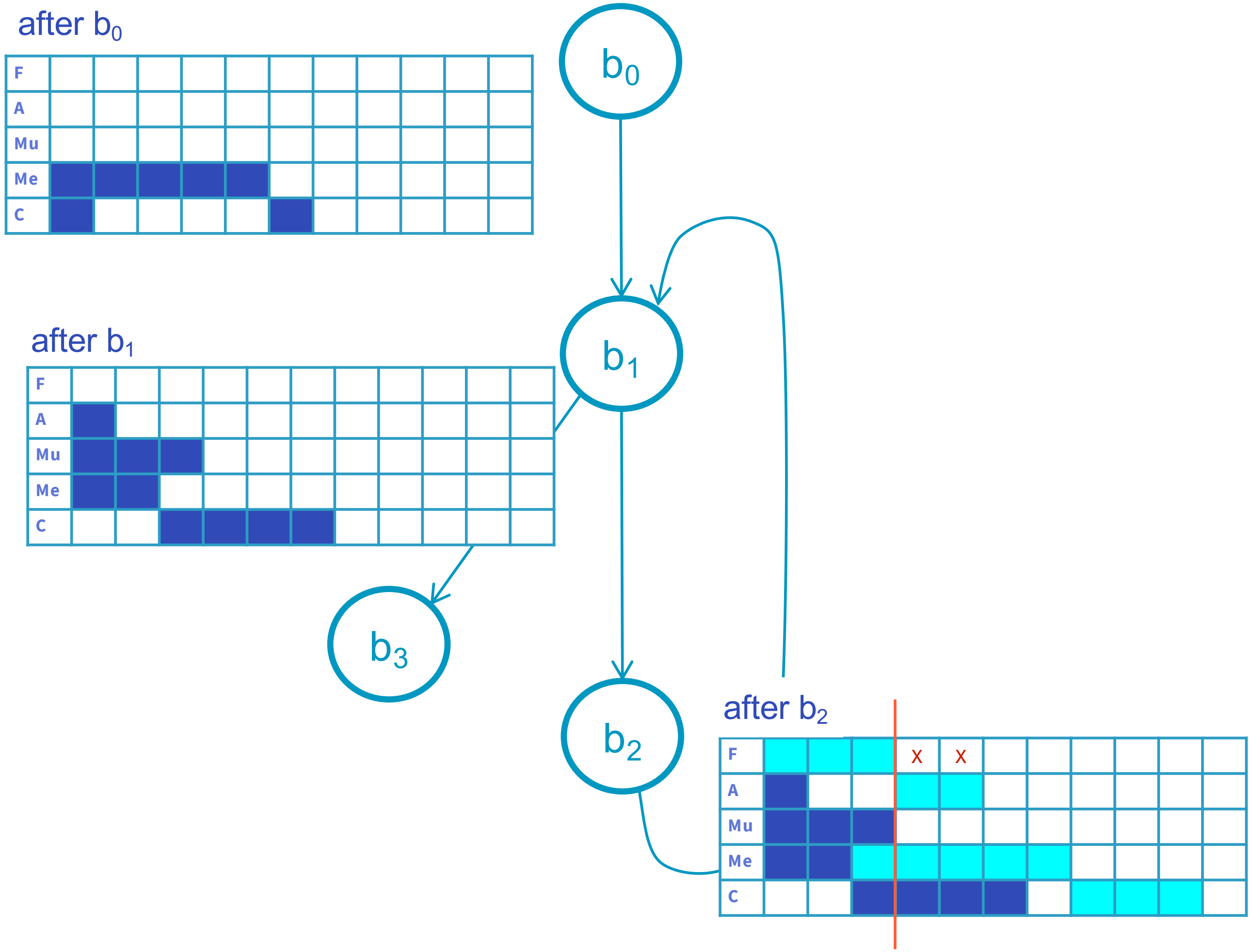



$t_0 = 11$

$t_{0\text{-}1} = 3$

# A sketch of pipeline analysis by abstract interpretation

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```

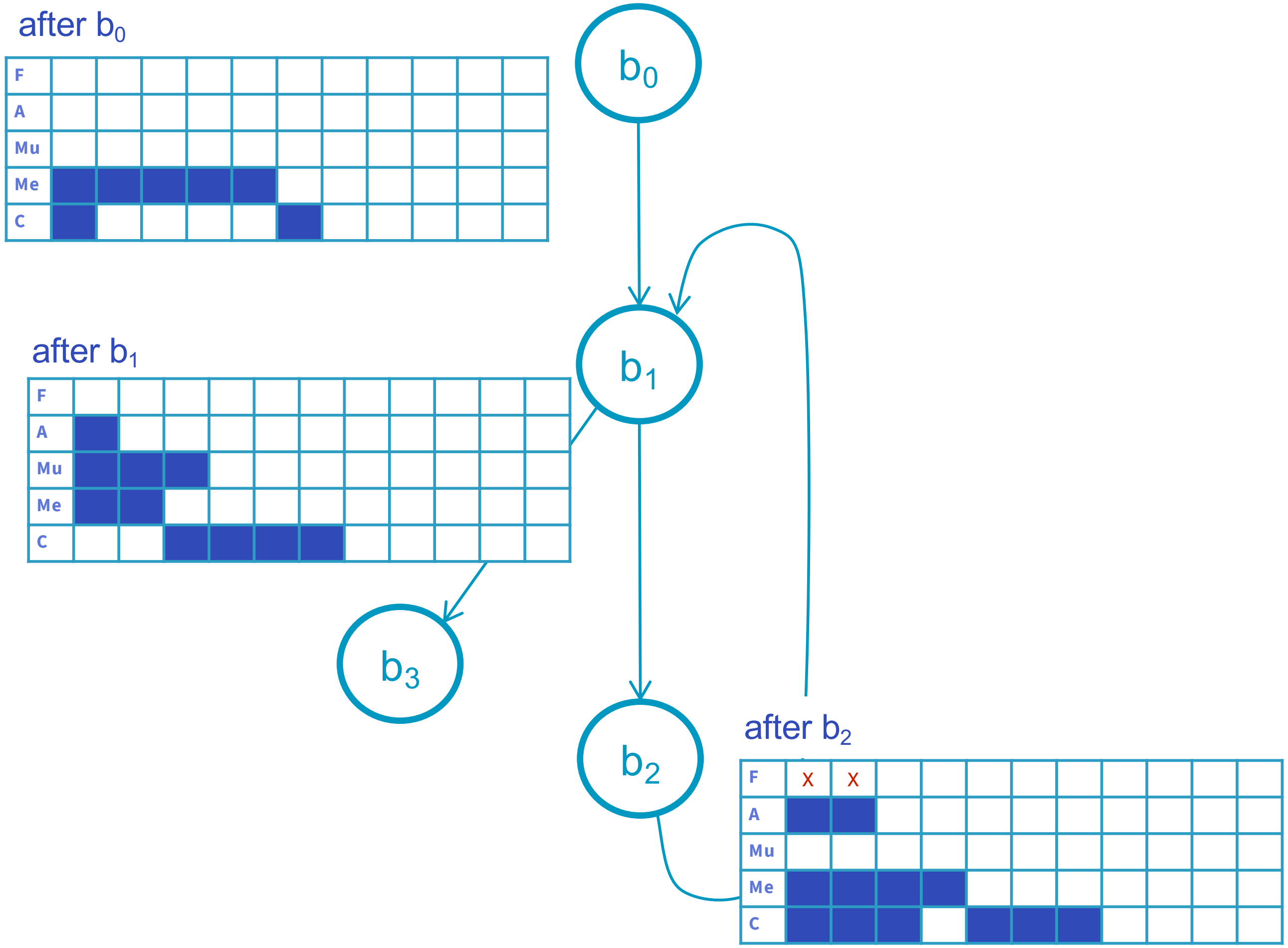



$t_0 = 11$

$t_{0\text{-}1} = 3$

# A sketch of pipeline analysis by abstract interpretation

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```

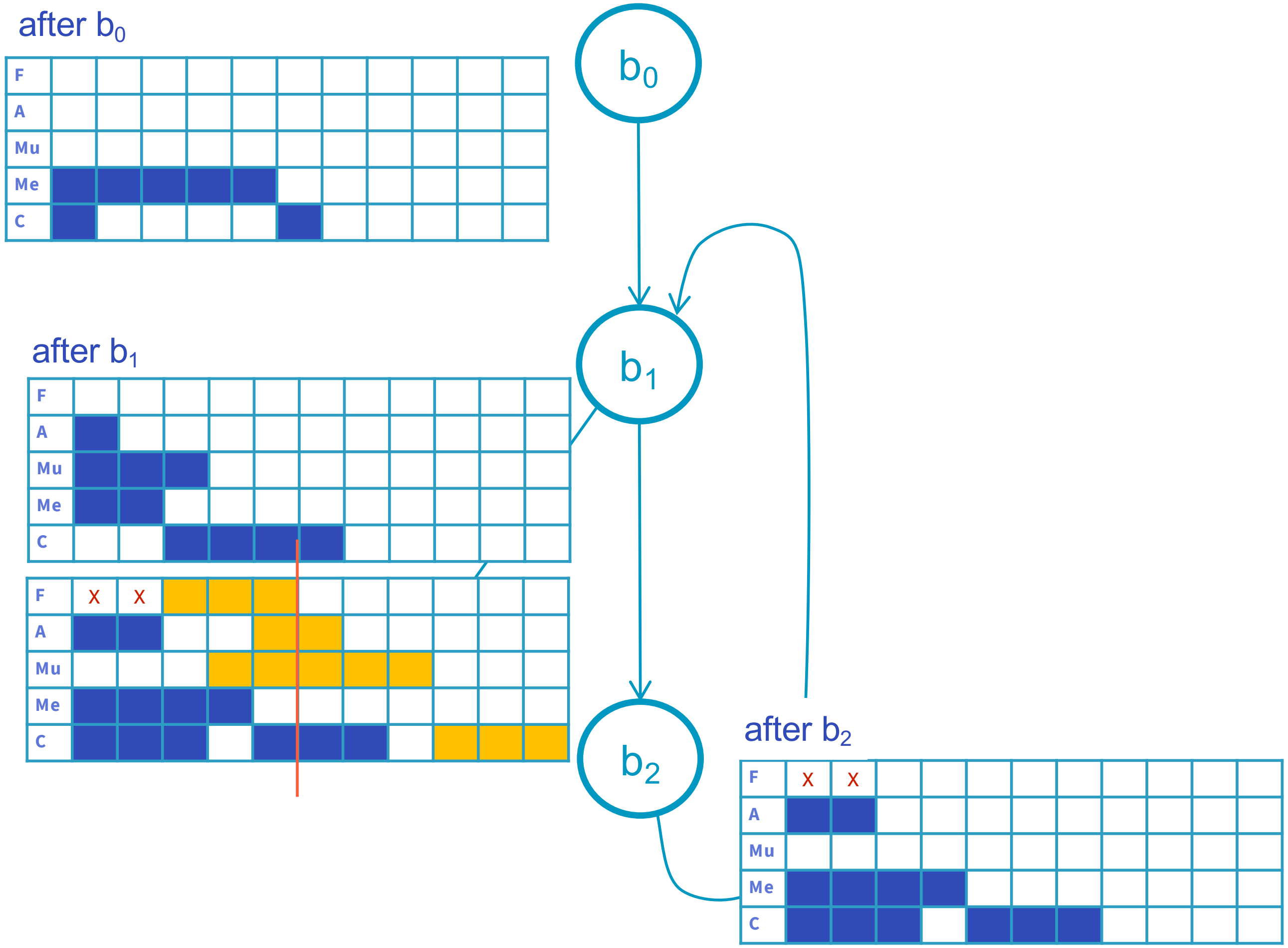



$t_0 = 11$

$t_{0\text{-}1} = 3$

# A sketch of pipeline analysis by abstract interpretation

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```




$t_0 = 11$

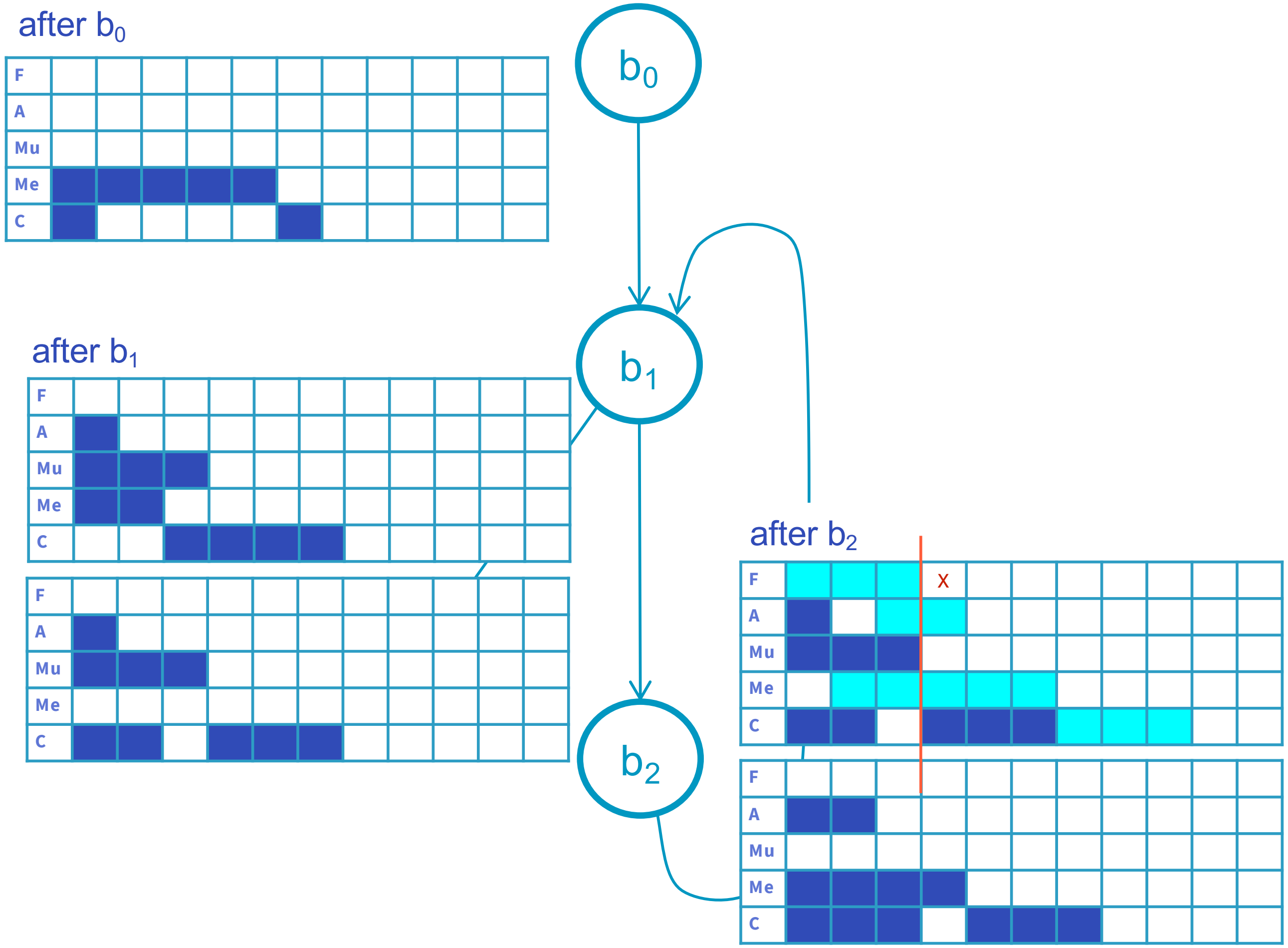
$t_{0\text{-}1} = 3$

# A sketch of pipeline analysis by abstract interpretation

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```

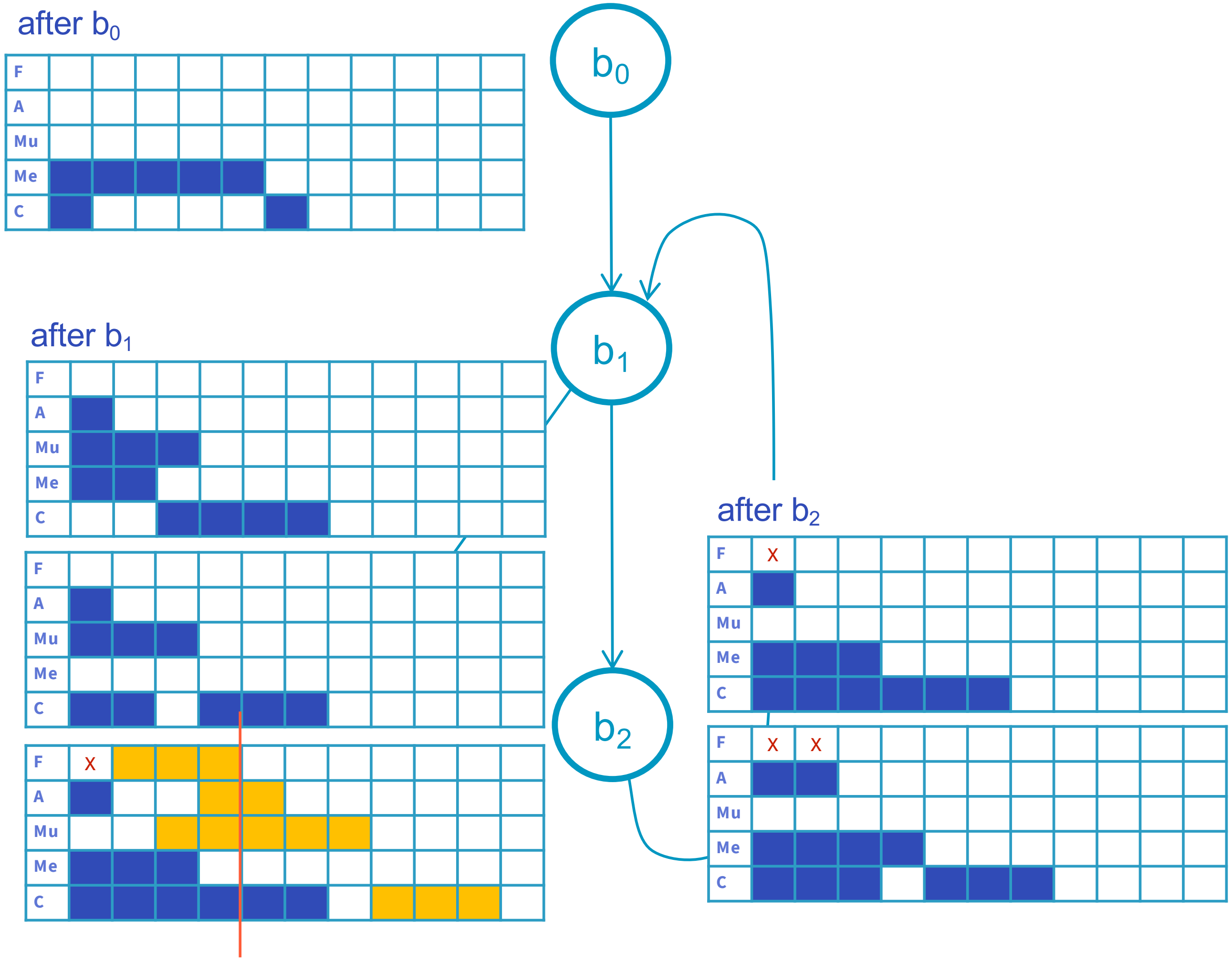
$t_0 = 11$

$t_{0\text{-}1} = 3$



# A sketch of pipeline analysis by abstract interpretation

```
0a   __start:   mov r0,#1
0b              mov r1,#0
0c              adr r2,a
0d              adr r3,init_val
0e              ldr r3,[r3]
1a   while:     mul r0,r1,r0
1b              cmp r1,#N
1c              bhs end
2a              str r3,[r2],#4
2b              add r1,r1,#1
2c              b while
3a   end:       adr r2,product
3b              str r0,[r2]
```

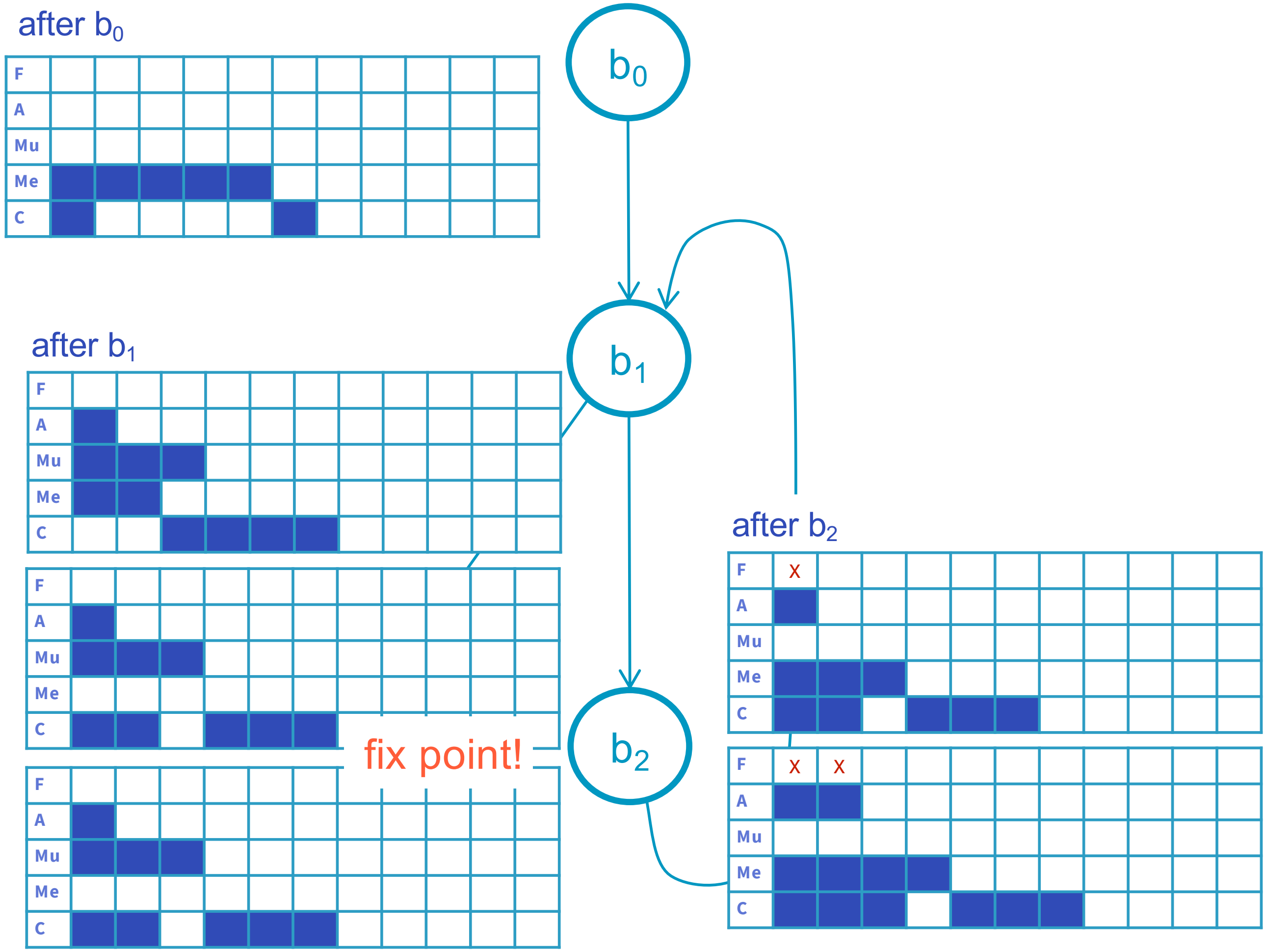



$t_0 = 11$

$t_{0-1} = 3$

# A sketch of pipeline analysis by abstract interpretation

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```

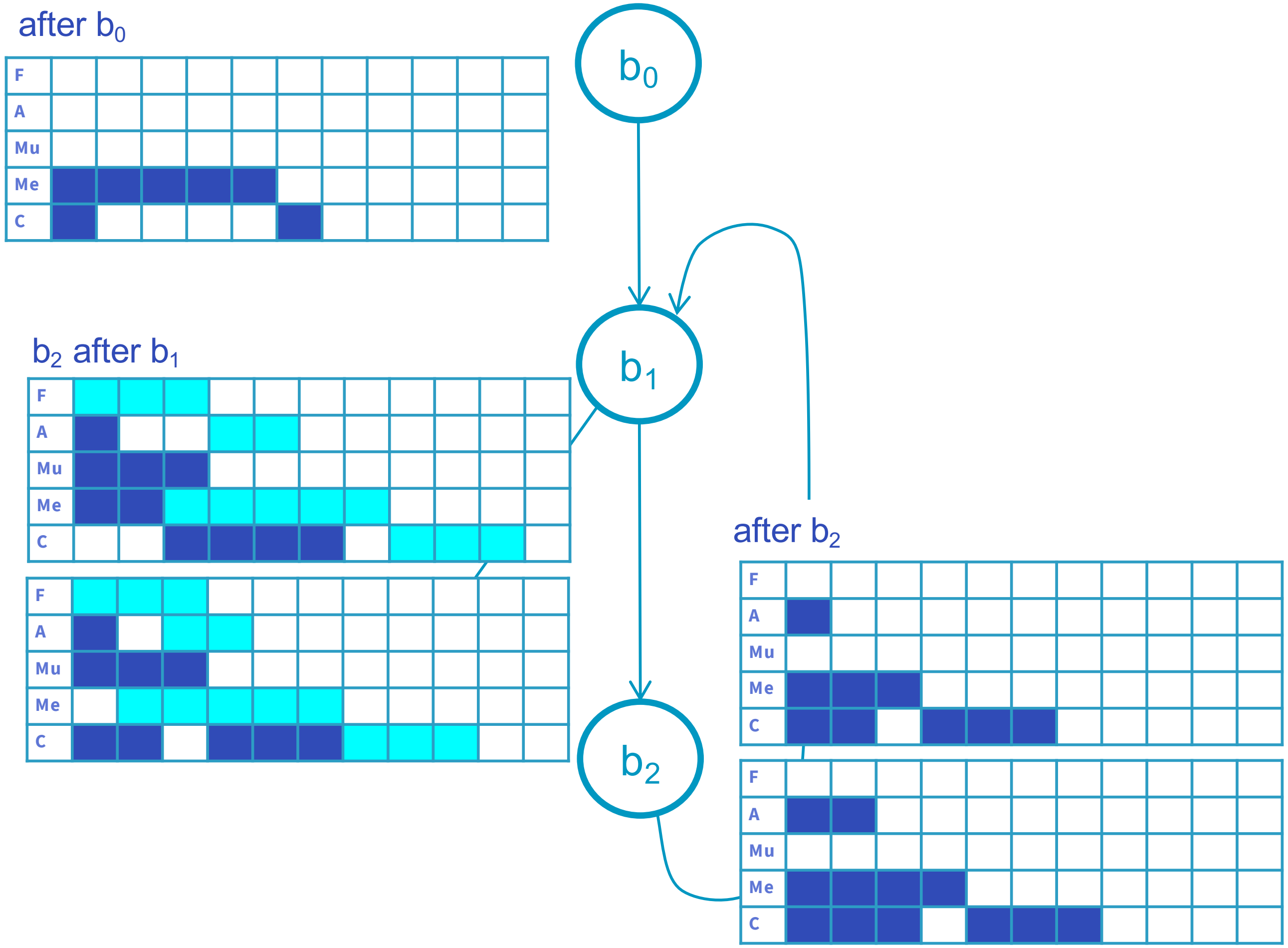



$t_0 = 11$

$t_{0\text{-}1} = 3$

# A sketch of pipeline analysis by abstract interpretation

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```



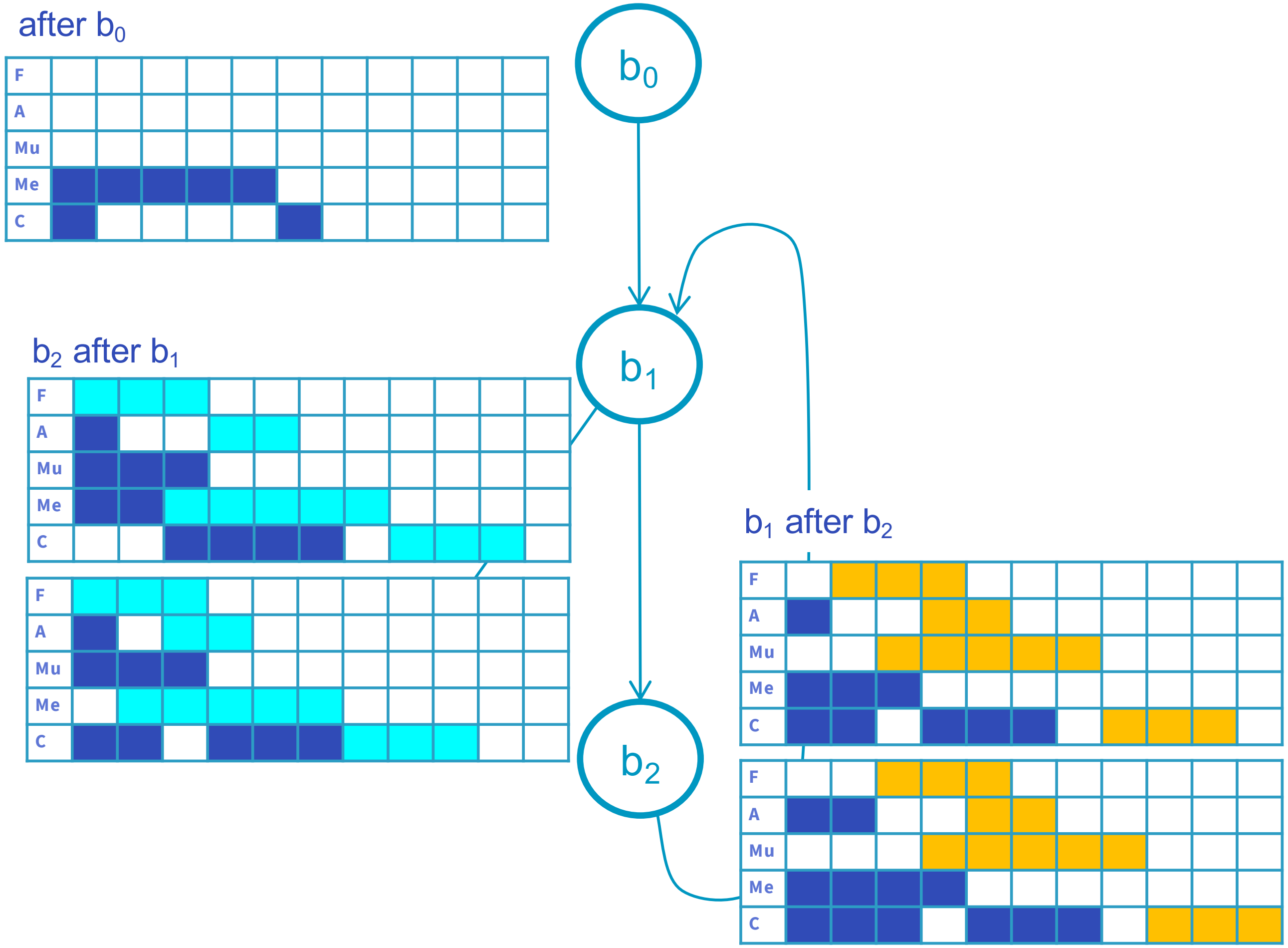

$t_0 = 11$

$t_{0\text{-}1} = 3$

$t_{1\text{-}2} = \max(4,3) = 4$

# A sketch of pipeline analysis by abstract interpretation

```
0a   __start:   mov r0,#1
0b              mov r1,#0
0c              adr r2,a
0d              adr r3,init_val
0e              ldr r3,[r3]
1a   while:     mul r0,r1,r0
1b              cmp r1,#N
1c              bhs end
2a              str r3,[r2],#4
2b              add r1,r1,#1
2c              b while
3a   end:       adr r2,product
3b              str r0,[r2]
```




$t_0 = 11$

$t_{0\text{-}1} = 3$

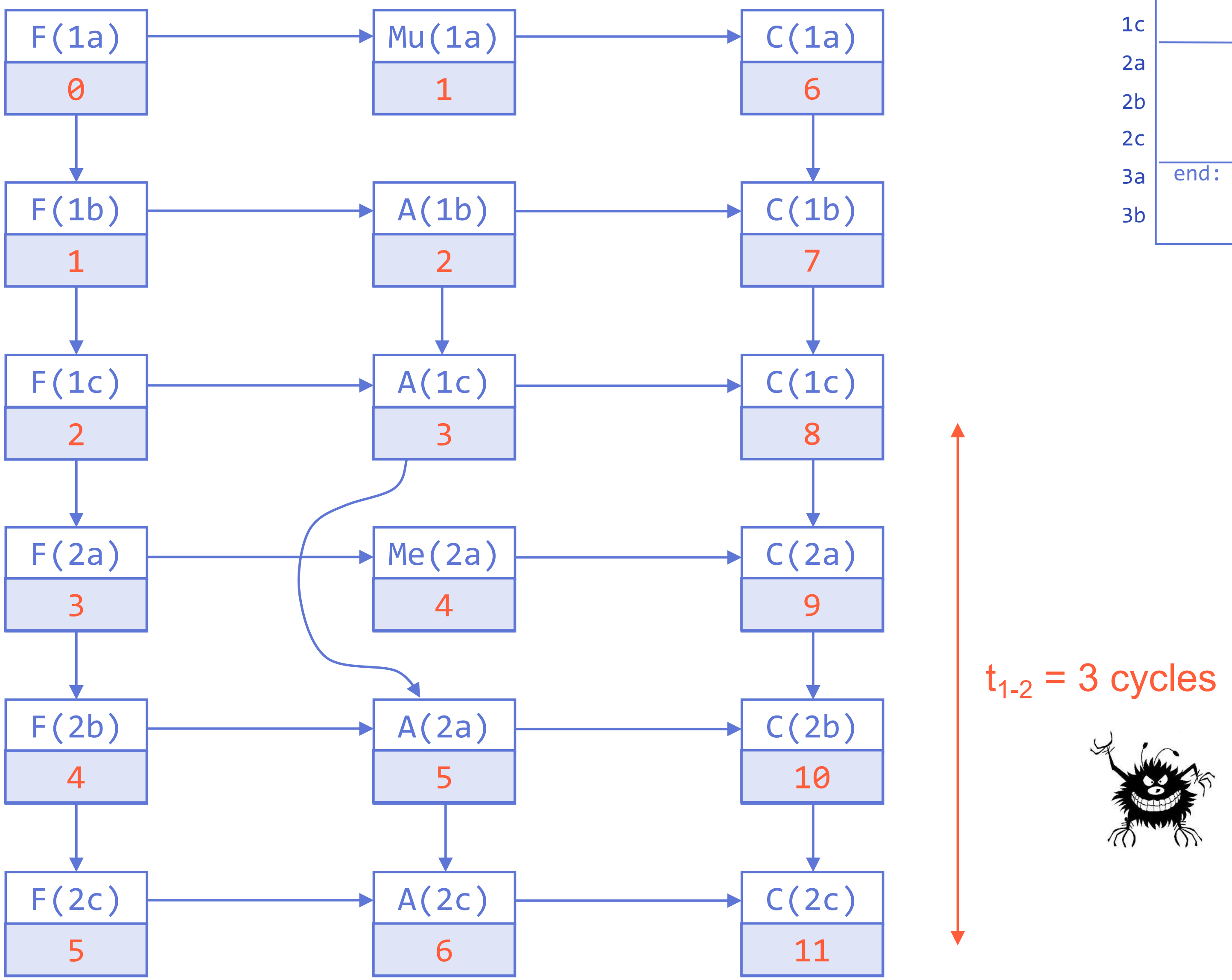
$t_{1\text{-}2} = \max(4,3) = 4$

$t_{2\text{-}1} = \max(4,4) = 4$

$t_{0\text{-}1\text{-}2} = 11 + 3 + 4 = 18$

# "Local" pipeline analysis based on execution graphs

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```



# "Local" pipeline analysis based on execution graphs

## Parameterized initial state

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```

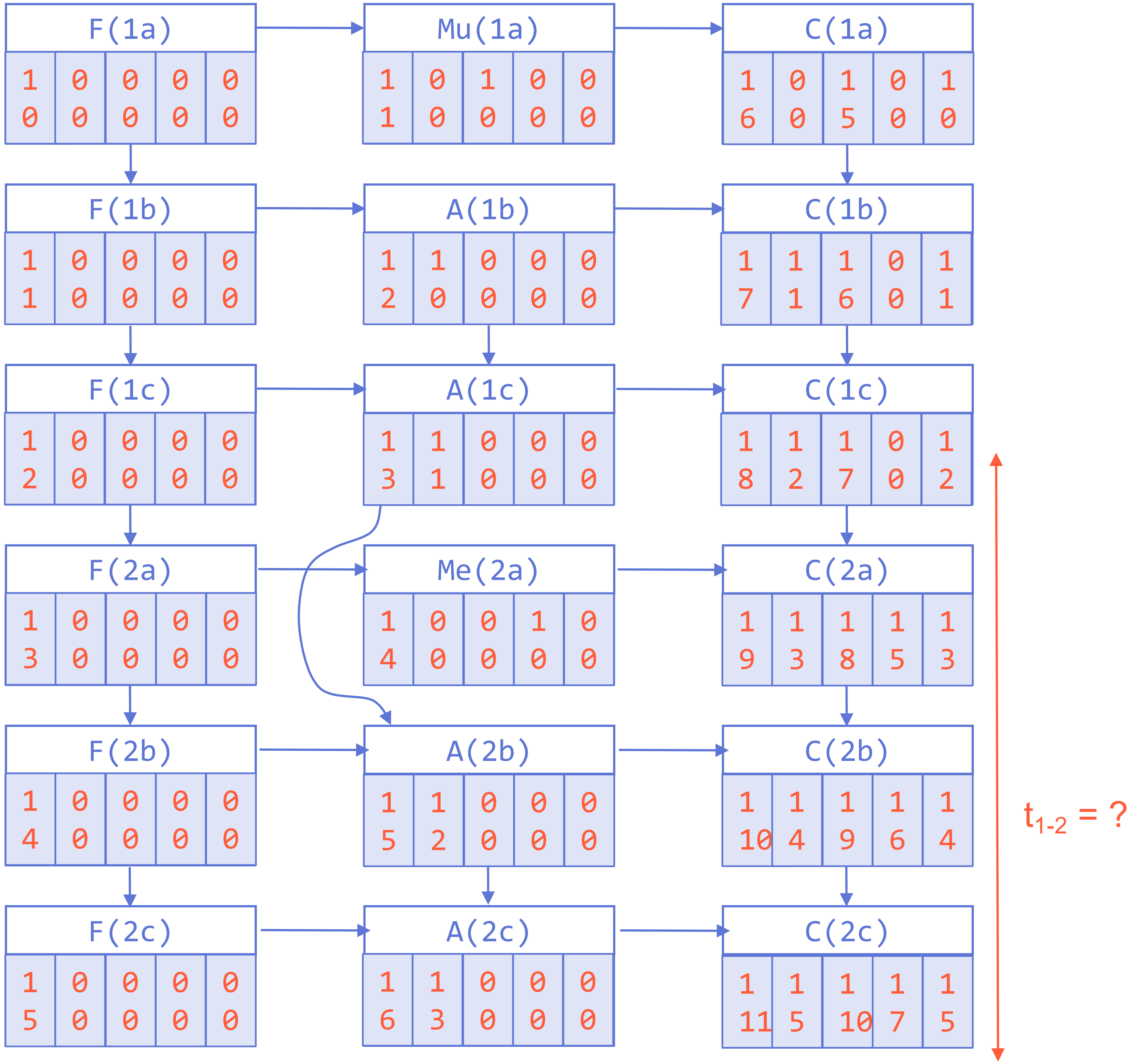
each resource has an availability date

starting date of a node

| F | A | Mu | Me | C | r0 | r1 | r2 | r3 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |

$e_i$ = depends on resource?

$d_i$ = delay after resource becomes available




```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```

| F | A | Mu | Me | C |
|---|---|---|---|---|
| | | | | |

# "Local" pipeline analysis based on execution graphs

| C(1c) | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 |
| 8 | 2 | 7 | 0 | 2 |

$t_{C(1c)} = \max(\ t_F+8\ ,\ \ t_A+2\ ,\ t_{Mu}+7\ \ ,\ \qquad\qquad ,\ t_C+2\ )$

$t_{Me} \leq t_C - 1$

| C(2c) | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 11 | 5 | 10 | 7 | 5 |

$t_{C(2c)} = \max(\ t_F+11\ ,\ t_A+5\ ,\ t_{Mu}+10\ ,\ \ t_{Me}+7\ ,\ t_C+5\ )$

$\leq t_C + 6$

| F | A | Mu | Me | C |
|---|---|---|---|---|
| | | | | |

$t_{C(2c)} - t_{C(1c)} = \max(\ 11\text{-}8\ ,\ \ 5\text{-}2\ ,\ 10\text{-}7\ \ ,\ \ 6\text{-}2\ ,\ 5\text{-}2\ )$

$t_{1\text{-}2} \leq 4$ cycles

# Handling variable instruction latencies

## Variable latencies?

- depending on operand values (e.g. multiplication)
- depending on the state of some components
  - example: instruction or data cache

## Timing anomalies

- the local worst-case might not be the global worst-case

# Timing anomalies

decode cycle

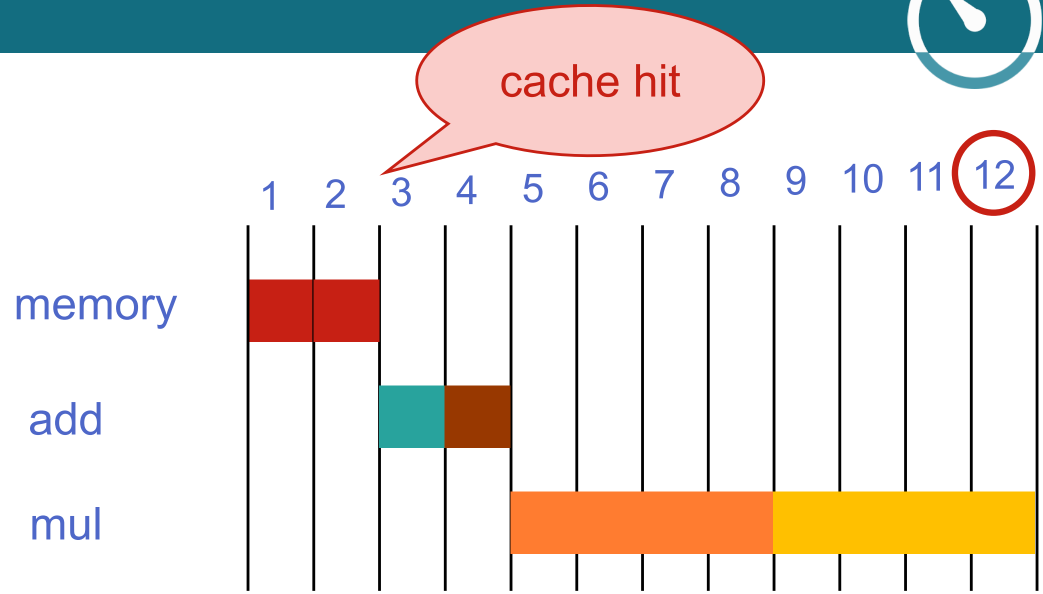
1 ldr r4,[r3]

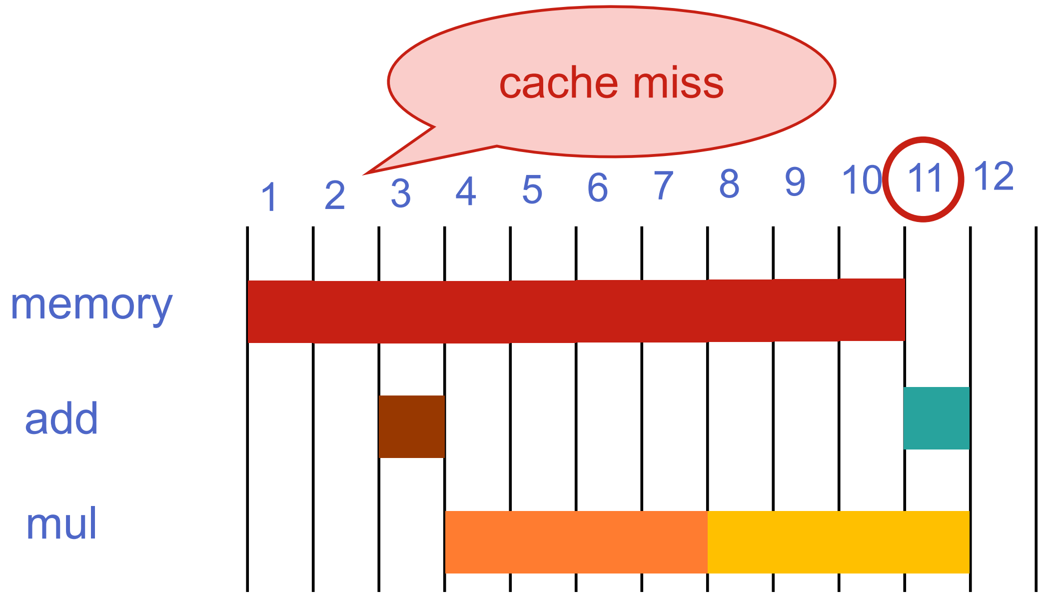
2 add r5,r4,r4

3 add r11,r10,r10

4 mul r11,r8,r11

5 mul r19,r1,r2



# Impact of timing anomalies

How can we get safe WCET estimations?

- by considering all possible latency values, one by one
  - additional states
  - additional cost values for basic blocks
  - changes in the ILP formulation?

- by considering the latency as a parameter when determining the cost of a basic block ?