NICTA

Timing Analysis of Binary Programs with UPPAAL

Franck Cassez
NICTA, Australia

Jean-Luc Béchennec
CNRS, France

ACSD 2013, Barcelona, July 2013

Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

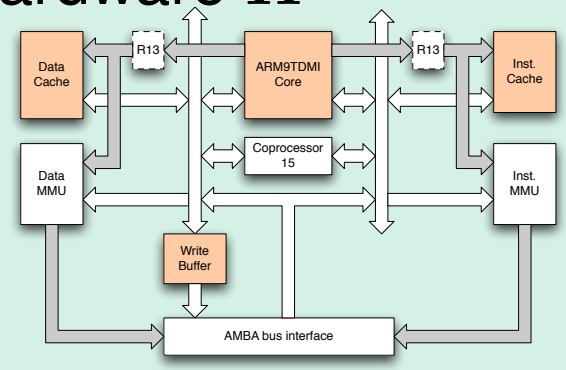
NICTA Members
NICTA Partners

The WCET Problem

Program P

```
10:  e3a03000    mov   r3, #0
14:  e58d3014    str   r3, [sp, #20]
18:  e3a03002    mov   r3, #2
1c:  e58d300c    str   r3, [sp, #12]
20:  ea00000a    b     50 <fib+0x50>
24:  e59d3010    ldr   r3, [sp, #16]
28:  e58d3018    str   r3, [sp, #24]
2c:  e59d2010    ldr   r2, [sp, #16]
30:  e59d3014    ldr   r3, [sp, #20]
34:  e0823003    add   r3, r2, r3
38:  e58d3010    str   r3, [sp, #16]
3c:  e59d3018    ldr   r3, [sp, #24]
40:  e58d3014    str   r3, [sp, #20]
44:  e59d300c    ldr   r3, [sp, #12]
48:  e2833001    add   r3, r3, #1
4c:  e58d300c    str   r3, [sp, #12]
50:  e59d200c    ldr   r2, [sp, #12]
54:  e59d3004    ldr   r3, [sp, #4]
58:  e1520003    cmp   r2, r3
5c:  daffffe0    ble   24 <fib+0x24>
```

Hardware H



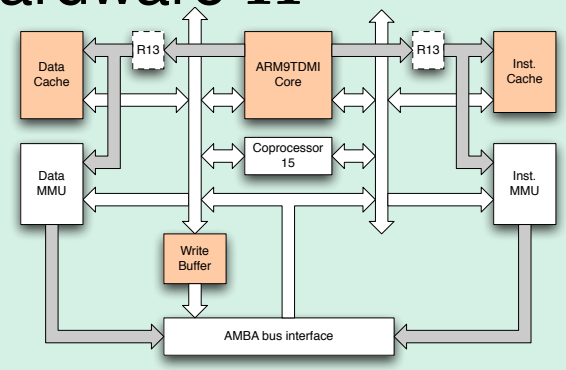
The WCET Problem

$d \in \mathcal{D}$ →

Program P

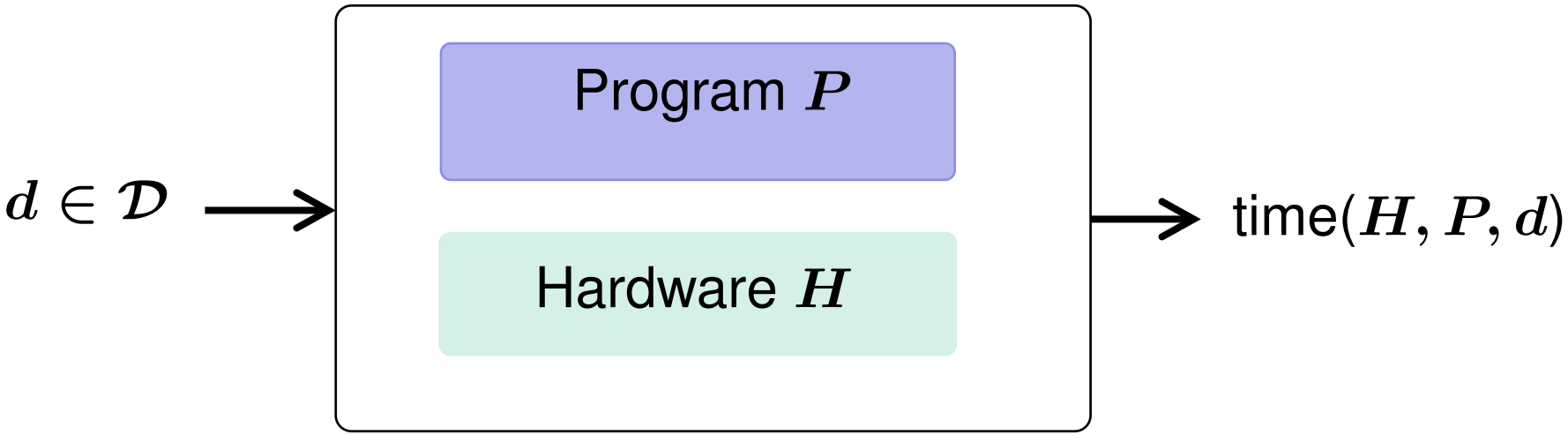
```
10:  e3a03000    mov   r3, #0
14:  e58d3014    str   r3, [sp, #20]
18:  e3a03002    mov   r3, #2
1c:  e58d300c    str   r3, [sp, #12]
20:  ea00000a    b     50 <fib+0x50>
24:  e59d3010    ldr   r3, [sp, #16]
28:  e58d3018    str   r3, [sp, #24]
2c:  e59d2010    ldr   r2, [sp, #16]
30:  e59d3014    ldr   r3, [sp, #20]
34:  e0823003    add   r3, r2, r3
38:  e58d3010    str   r3, [sp, #16]
3c:  e59d3018    ldr   r3, [sp, #24]
40:  e58d3014    str   r3, [sp, #20]
44:  e59d300c    ldr   r3, [sp, #12]
48:  e2833001    add   r3, r3, #1
4c:  e58d300c    str   r3, [sp, #12]
50:  e59d200c    ldr   r2, [sp, #12]
54:  e59d3004    ldr   r3, [sp, #4]
58:  e1520003    cmp   r2, r3
5c:  daffffe0    ble   24 <fib+0x24>
```

Hardware H

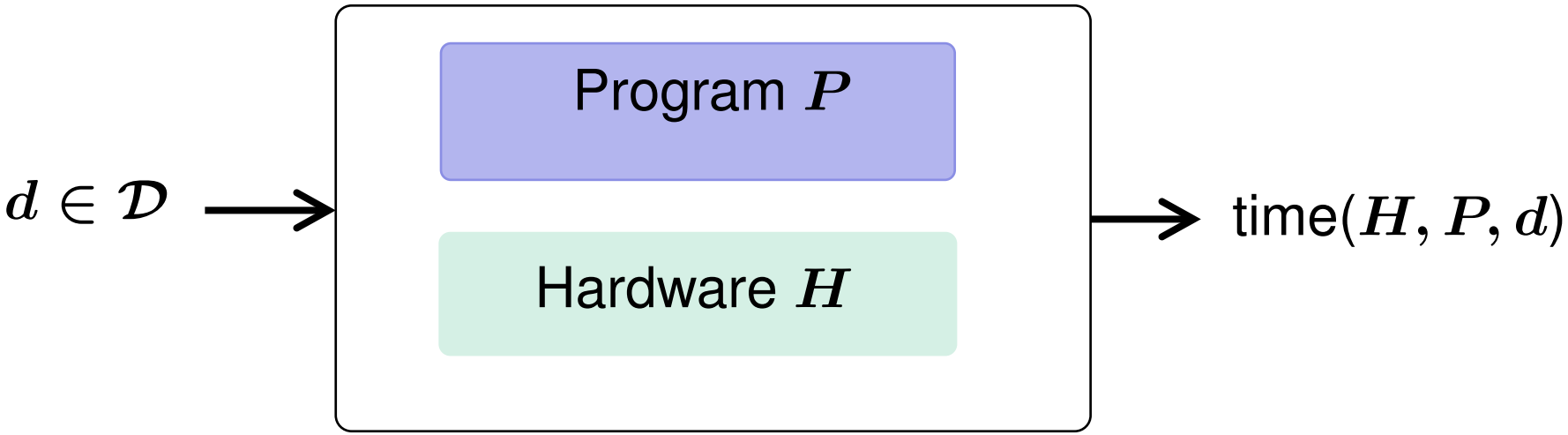


→ time(H, P, d)

The WCET Problem

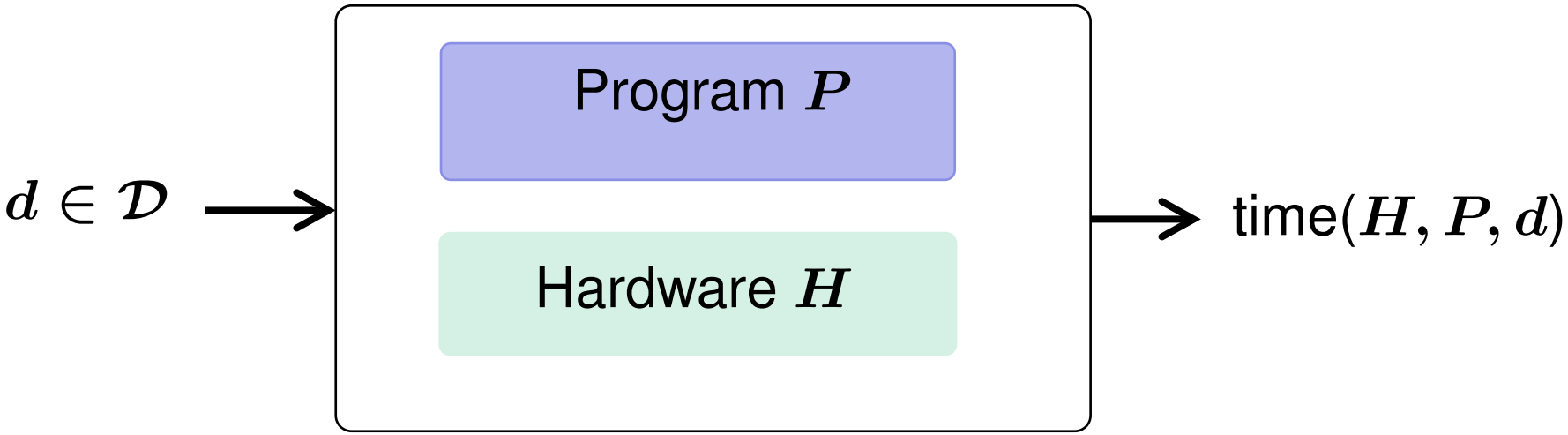


The WCET Problem



$$\text{WCET}(H, P) = \max_{d \in \mathcal{D}} \text{time}(H, P, d)$$

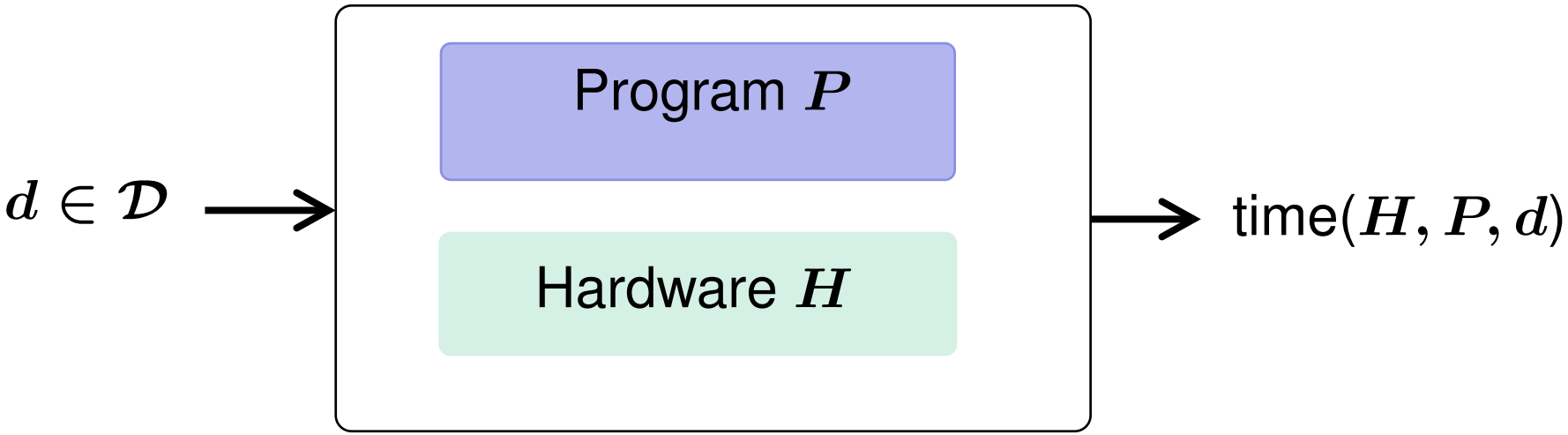
The WCET Problem



$$\text{WCET}(H, P) = \max_{d \in \mathcal{D}} \text{time}(H, P, d)$$

$$\text{WCET}(H, P) \leq \quad \text{WCET-UB}(H, P)$$

The WCET Problem



$$\text{WCET}(H, P) = \max_{d \in \mathcal{D}} \text{time}(H, P, d)$$

$$\text{WCET}(H, P) \leq \quad \text{WCET-UB}(H, P) \leq \quad (1+\varepsilon) \times \text{WCET}(H, P)$$

Related Work & Existing Tools

NICTA

- Tests & Simulation
  - Random, probabilistic
  - Real board or simulator
  - Non exhaustive
  - Tools: RapiTime (pWCET, mTime)

- Abstract Interpretation & Integer Linear Programming
  - Compute a CFG
  - Determine loop bounds
  - Build a weighted graph
  - Solve an ILP
  - Tools: Bound-T, OTAWA, TuBound, aiT (Absint)

Related Work & Existing Tools

NICTA

- Tests & Simulation
  - Random, probabilistic
  - Real board or simulator
  - Non exhaustive
  - Tools: RapiTime (pWCET, mTime)

Lower Bound
Unsafe

Easy to implement

- Abstract Interpretation & Integer Linear Programming
  - Compute a CFG
  - Determine loop bounds
  - Build a weighted graph
  - Solve an ILP
  - Tools: Bound-T, OTAWA, TuBound, aiT (Absint)

Upper Bound
safe

Manual Annotations
Over-pessimistic
Monolithic

What about Model-Checking Based Techniques?

NICTA

Bad News

Wilhelm, R.: Why AI+ILP is good for WCET, but MC is not, nor ILP alone.

VMCAI, 2004

What about Model-Checking Based Techniques?

NICTA

Bad News

Wilhelm, R.: Why AI+ILP is good for WCET, but MC is not, nor ILP alone.

VMCAI, 2004

Good News

Metzner, A.: Why model checking can improve WCET analysis.

CAV, 2004

Dalsgaard, A.E., Olesen, M.C., Toft, M.: Modular execution time analysis using model checking.

Master's thesis, Dept. of Computer Science, Aalborg University, DK (2009)

Model-Checking Based Techniques

NICTA

METAMOC

- Modular Execution Time Analysis using Model Checking
- Timed Automata to model Hardware
- Loop bounds, value analysis to compute CFG
- Loop unfolding
- UPPAAL to compute longest path (WCET)

Our Methodology

- Automatic computation of WCET
- No loop bounds computation
- Follow up of Timed games for computing WCET for pipelined processors with caches. In *ACSD'2011.*
- UPPAAL to compute longest path (WCET)

Our Contribution

NICTA

- Fully automatic computation of WCET
- Modular Technique
  - Build CFG
  - Build model of hardware
  - Model-check product of CFG and Hardware
- Formal Hardware model for ARM920T
- Comparison of computed WCETs and measured WCETs

Overview

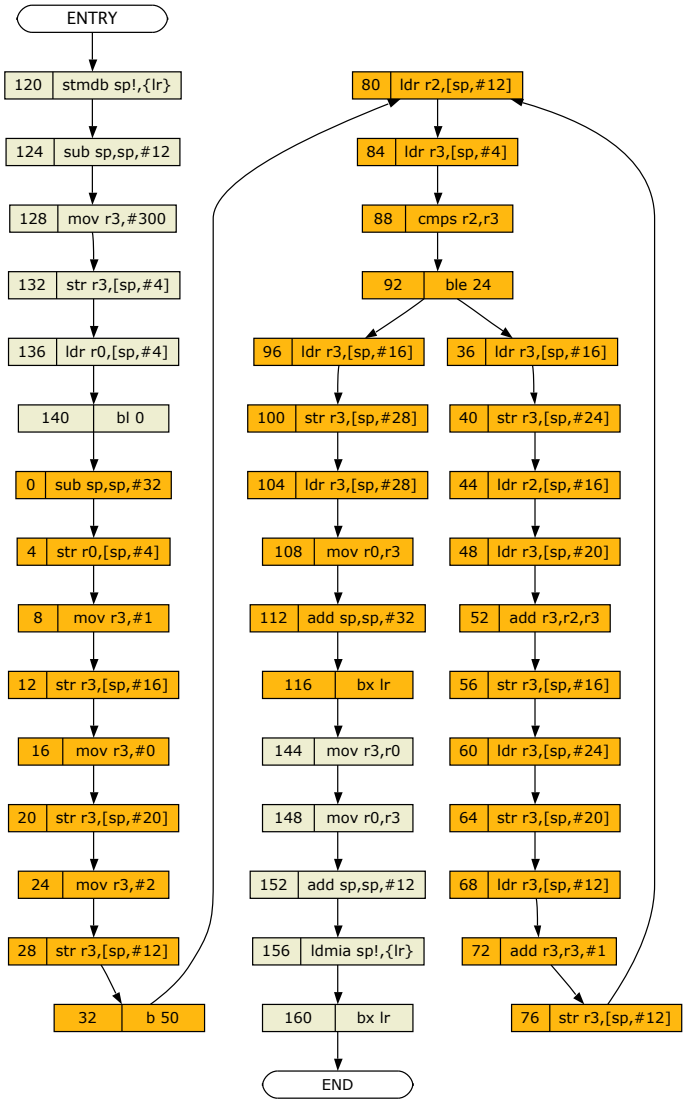
- What type of programs and architectures?
- Modular Computation of WCET
- CFG Reconstruction
- Hardware Formal Models for ARM920T
- Experiments
- Conclusion

Binary Program: Fibonacci

```
00000000 <fib>:
   0:  e24dd020    sub   sp, sp, #32
   4:  e58d0004    str   r0, [sp, #4]
   8:  e3a03001    mov   r3, #1
   c:  e58d3010    str   r3, [sp, #16]
  10:  e3a03000    mov   r3, #0
  14:  e58d3014    str   r3, [sp, #20]
  18:  e3a03002    mov   r3, #2
  1c:  e58d300c    str   r3, [sp, #12]
  20:  ea00000a    b     50 <fib+0x50>
  24:  e59d3010    ldr   r3, [sp, #16]
  28:  e58d3018    str   r3, [sp, #24]
  2c:  e59d2010    ldr   r2, [sp, #16]
  30:  e59d3014    ldr   r3, [sp, #20]
  34:  e0823003    add   r3, r2, r3
  38:  e58d3010    str   r3, [sp, #16]
  3c:  e59d3018    ldr   r3, [sp, #24]
  40:  e58d3014    str   r3, [sp, #20]
  44:  e59d300c    ldr   r3, [sp, #12]
  48:  e2833001    add   r3, r3, #1
  4c:  e58d300c    str   r3, [sp, #12]
  50:  e59d200c    ldr   r2, [sp, #12]
  54:  e59d3004    ldr   r3, [sp, #4]
  58:  e1520003    cmp   r2, r3
  5c:  dafffff0    ble   24 <fib+0x24>
  60:  e59d3010    ldr   r3, [sp, #16]
  64:  e58d301c    str   r3, [sp, #28]
  68:  e59d301c    ldr   r3, [sp, #28]
  6c:  e1a00003    mov   r0, r3
  70:  e28dd020    add   sp, sp, #32
  74:  e12fff1e    bx    lr

00000078 <main>:
  78:  e52de004    push  {lr}
  7c:  e24dd00c    sub   sp, sp, #12
  80:  e3a03f4b    mov   r3, #300
  84:  e58d3004    str   r3, [sp, #4]
  88:  e59d0004    ldr   r0, [sp, #4]
  8c:  ebffffdb    bl    0 <fib>
  90:  e1a03000    mov   r3, r0
  94:  e1a00003    mov   r0, r3
  98:  e28dd00c    add   sp, sp, #12
  9c:  e49de004    pop   {lr}
  a0:  e12fff1e    bx    lr
```

Binary Program: Fibonacci

```
00000000 <fib>:
   0:  e24dd020     sub   sp, sp, #32
   4:  e58d0004     str   r0, [sp, #4]
   8:  e3a03001     mov   r3, #1
   c:  e58d3010     str   r3, [sp, #16]
  10:  e3a03000     mov   r3, #0
  14:  e58d3014     str   r3, [sp, #20]
  18:  e3a03002     mov   r3, #2
  1c:  e58d300c     str   r3, [sp, #12]
  20:  ea00000a     b     50 <fib+0x50>
  24:  e59d3010     ldr   r3, [sp, #16]
  28:  e58d3018     str   r3, [sp, #24]
  2c:  e59d2010     ldr   r2, [sp, #16]
  30:  e59d3014     ldr   r3, [sp, #20]
  34:  e0823003     add   r3, r2, r3
  38:  e58d3010     str   r3, [sp, #16]
  3c:  e59d3018     ldr   r3, [sp, #24]
  40:  e58d3014     str   r3, [sp, #20]
  44:  e59d300c     ldr   r3, [sp, #12]
  48:  e2833001     add   r3, r3, #1
  4c:  e58d300c     str   r3, [sp, #12]
  50:  e59d200c     ldr   r2, [sp, #12]
  54:  e59d3004     ldr   r3, [sp, #4]
  58:  e1520003     cmp   r2, r3
  5c:  dafffff0     ble   24 <fib+0x24>
  60:  e59d3010     ldr   r3, [sp, #16]
  64:  e58d301c     str   r3, [sp, #28]
  68:  e59d301c     ldr   r3, [sp, #28]
  6c:  e1a00003     mov   r0, r3
  70:  e28dd020     add   sp, sp, #32
  74:  e12fff1e     bx    lr

00000078 <main>:
  78:  e52de004     push  {lr}
  7c:  e24dd00c     sub   sp, sp, #12
  80:  e3a03f4b     mov   r3, #300
  84:  e58d3004     str   r3, [sp, #4]
  88:  e59d0004     ldr   r0, [sp, #4]
  8c:  ebfffffb     bl    0 <fib>
  90:  e1a03000     mov   r3, r0
  94:  e1a00003     mov   r0, r3
  98:  e28dd00c     add   sp, sp, #12
  9c:  e49de004     pop   {lr}
  a0:  e12fff1e     bx    lr
```



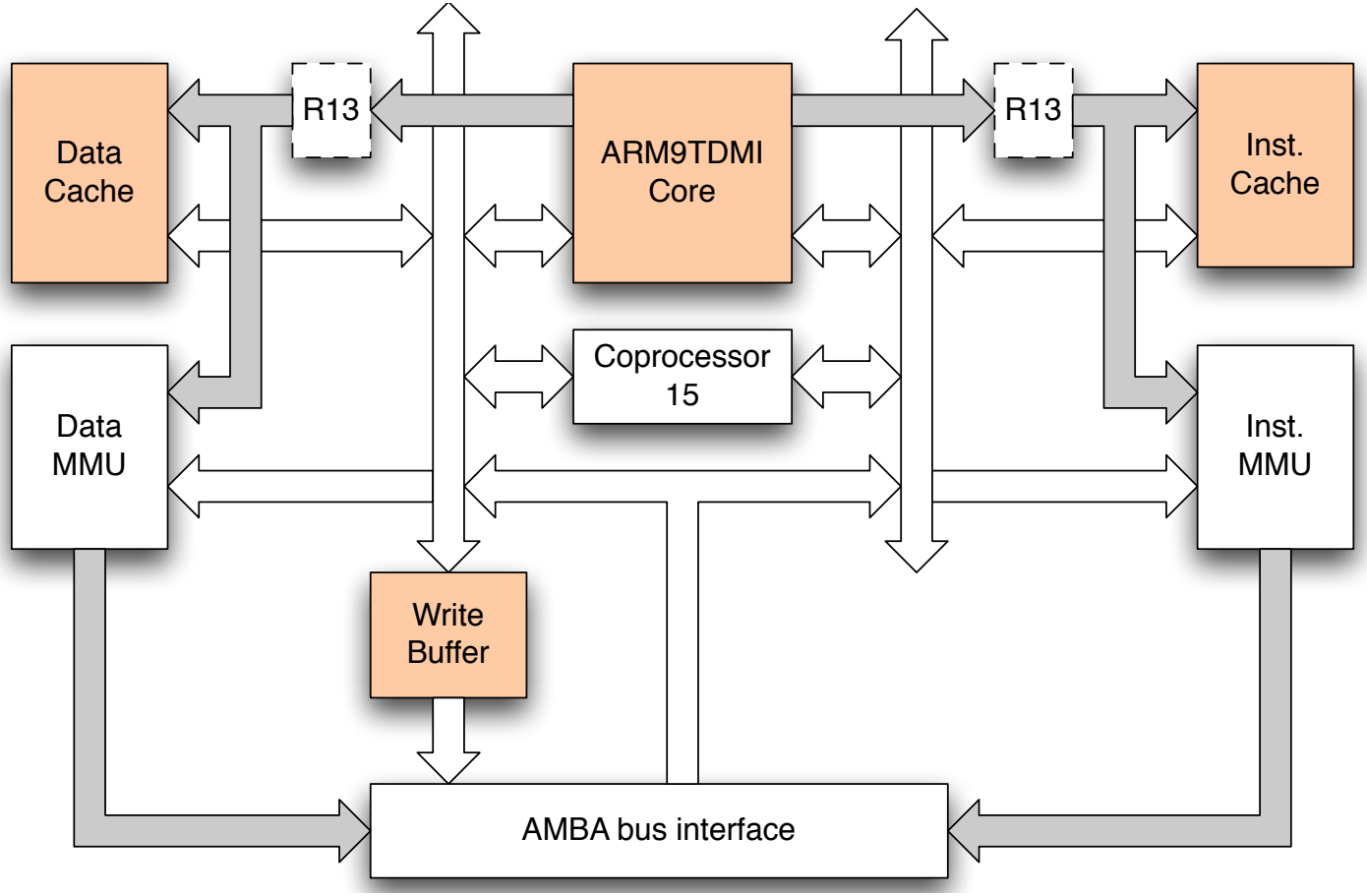
Example: Fibonacci Program

```
00000000 <fib>:
   0:  e24dd020     sub    sp, sp, #32
   4:  e58d0004     str    r0, [sp, #4]
   8:  e3a03001     mov    r3, #1
   c:  e58d3010     str    r3, [sp, #16]
  10:  e3a03000     mov    r3, #0
  14:  e58d3014     str    r3, [sp, #20]
  18:  e3a03002     mov    r3, #2
  1c:  e58d300c     str    r3, [sp, #12]
  20:  ea00000a     b      50 <fib+0x50>
  24:  e59d3010     ldr    r3, [sp, #16]
  28:  e58d3018     str    r3, [sp, #24]
  2c:  e59d2010     ldr    r2, [sp, #16]
  30:  e59d3014     ldr    r3, [sp, #20]
  34:  e0823003     add    r3, r2, r3
  38:  e58d3010     str    r3, [sp, #16]
  3c:  e59d3018     ldr    r3, [sp, #24]
  40:  e58d3014     str    r3, [sp, #20]
  44:  e59d300c     ldr    r3, [sp, #12]
  48:  e2833001     add    r3, r3, #1
  4c:  e58d300c     str    r3, [sp, #12]
  50:  e59d200c     ldr    r2, [sp, #12]
  54:  e59d3004     ldr    r3, [sp, #4]
  58:  e1520003     cmp    r2, r3
```

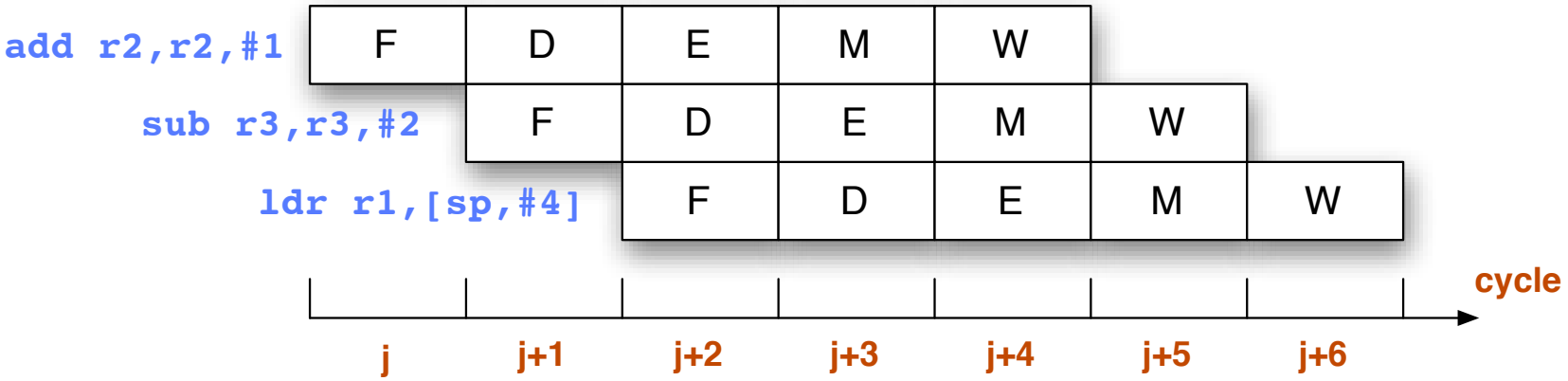
```
  60:  e59d3010     ldr    r3, [sp, #16]
  64:  e58d301c     str    r3, [sp, #28]
  68:  e59d301c     ldr    r3, [sp, #28]
  6c:  e1a00003     mov    r0, r3
  70:  e28dd020     add    sp, sp, #32
  74:  e12fff1e     bx     lr

00000078 <main>:
  78:  e52de004     push   {lr}
  7c:  e24dd00c     sub    sp, sp, #12
  80:  e3a03f4b     mov    r3, #300
  84:  e58d3004     str    r3, [sp, #4]
  88:  e59d0004     ldr    r0, [sp, #4]
  8c:  ebffffdb     bl     0 <fib>
  90:  e1a03000     mov    r3, r0
  94:  e1a00003     mov    r0, r3
  98:  e28dd00c     add    sp, sp, #12
  9c:  e49de004     pop    {lr}
  a0:  e12fff1e     bx     lr
```

Hardware: ARM920T

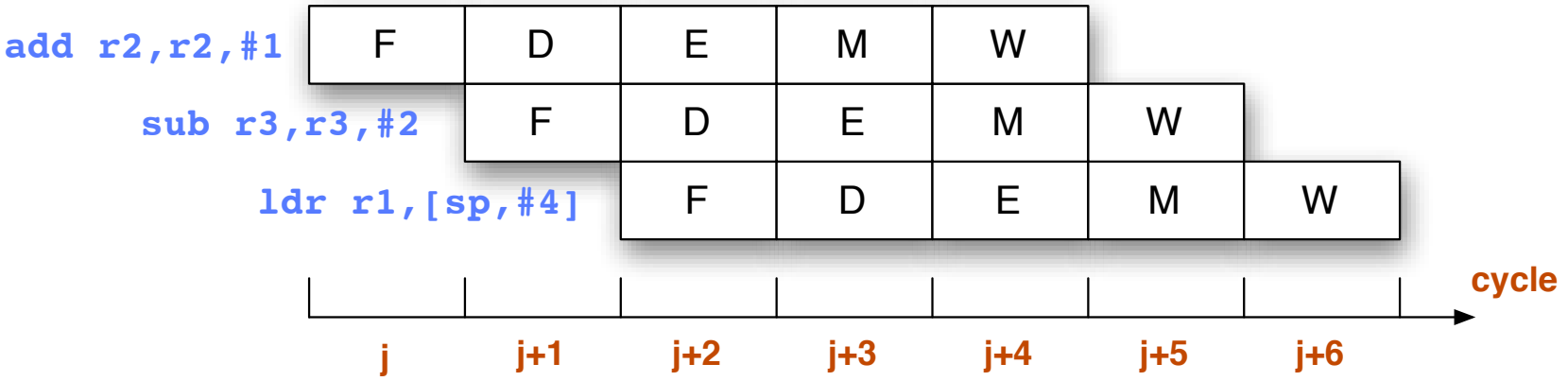
- RISC processor, 16 registers
- memory load/store and multiple ldr/str
- Data and Instruction Caches



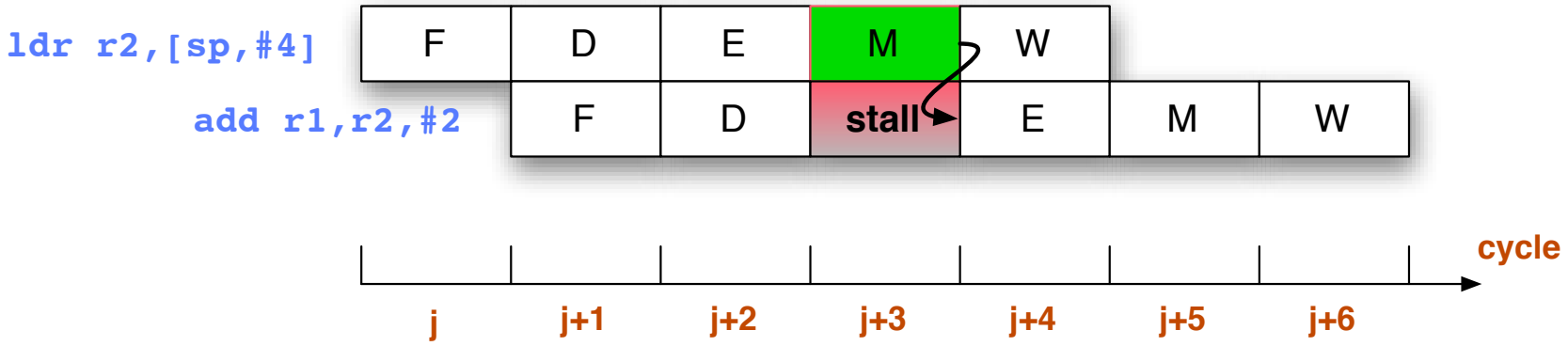
Pipelining and Pipeline Stalls



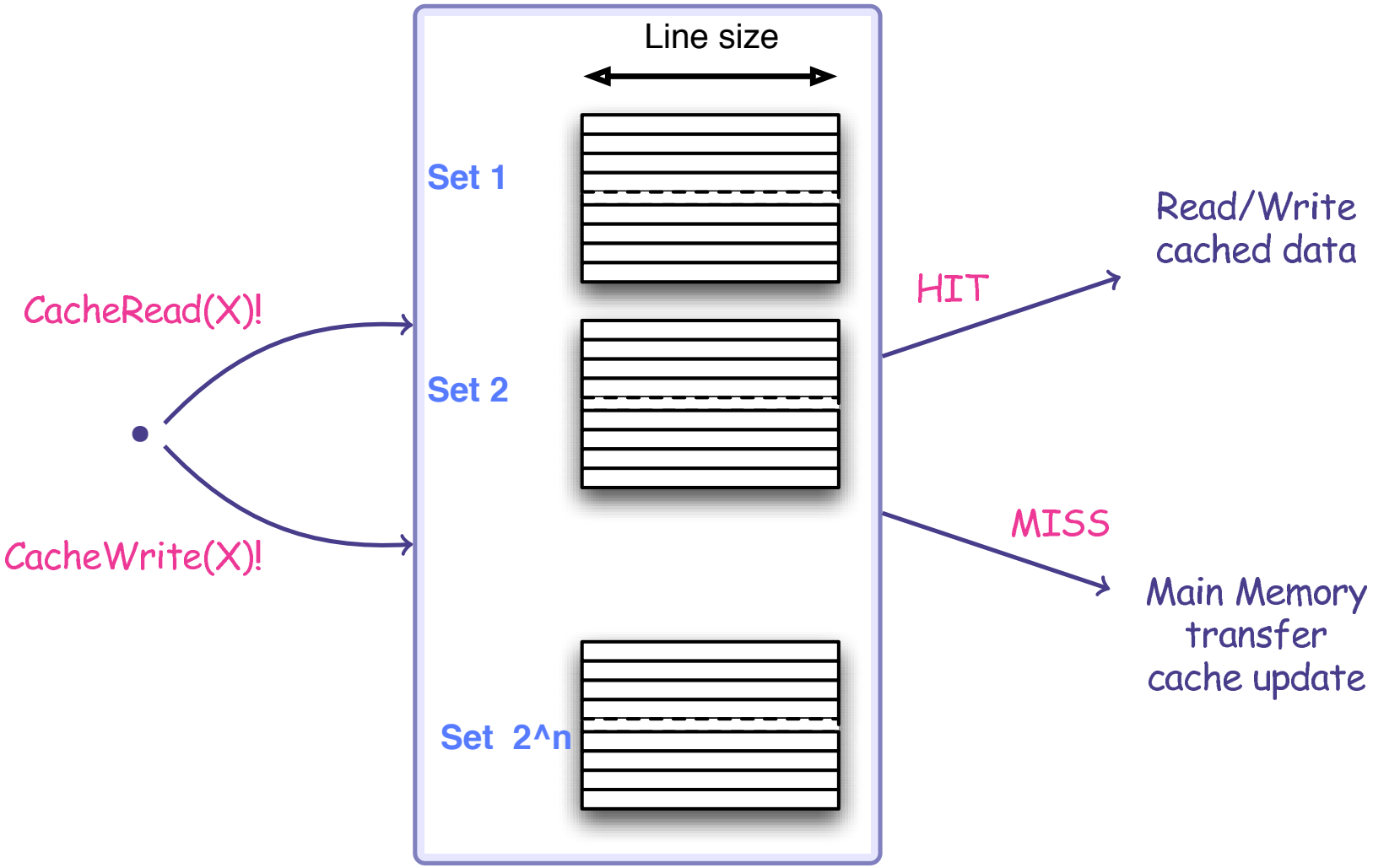
Pipelining and Pipeline Stalls



Stalls may occur!



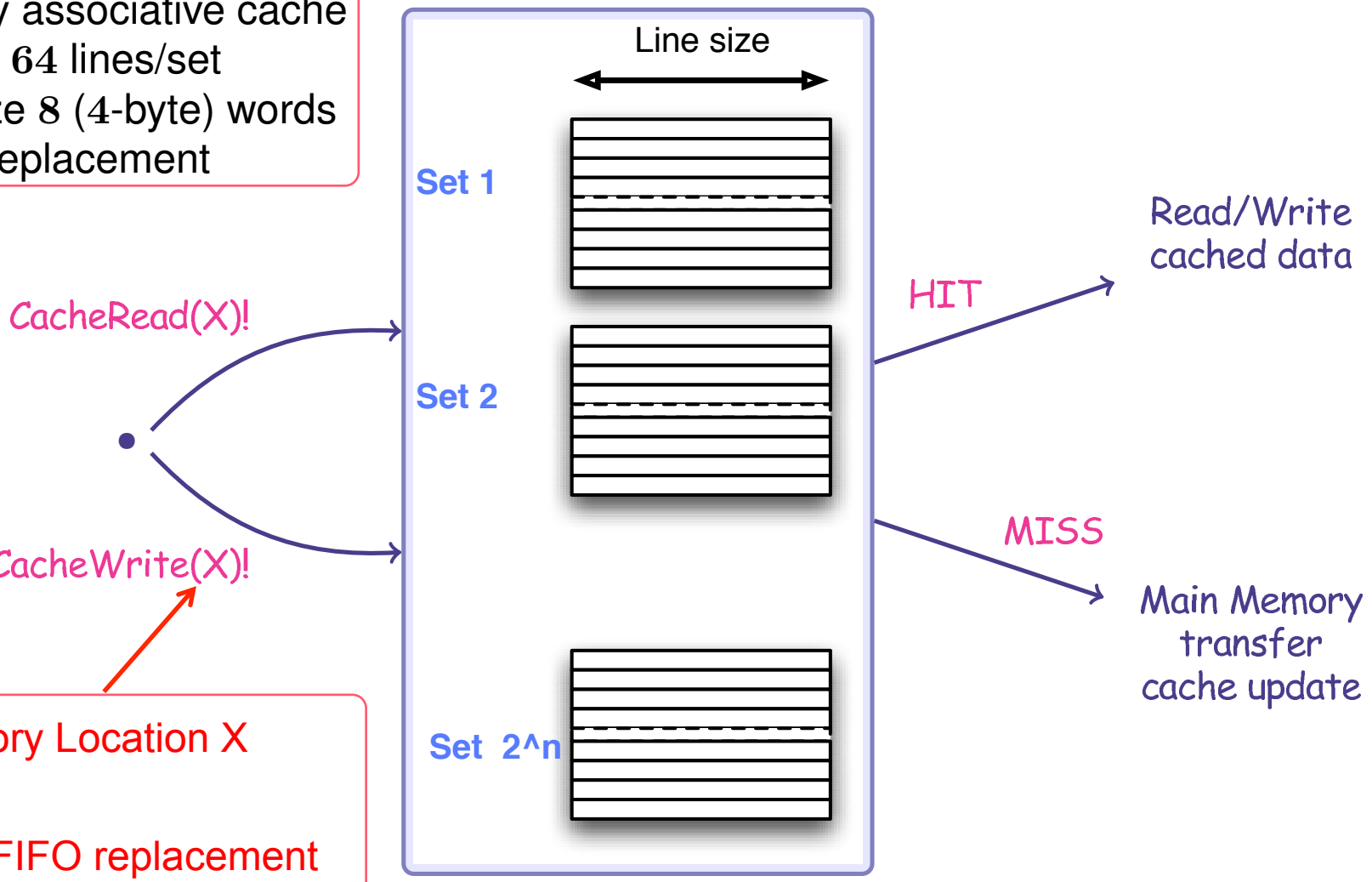
Caches



Caches

NICTA

64-way associative cache
8 sets, 64 lines/set
line size 8 (4-byte) words
FIFO replacement

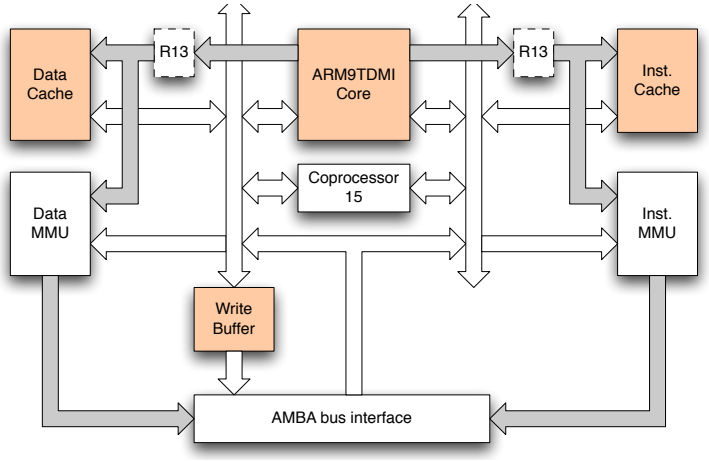


Memory Location X

Miss/FIFO replacement

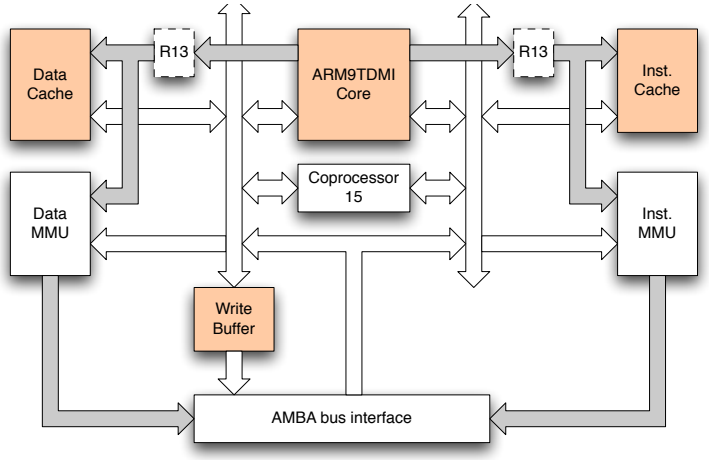
What do we Need to Compute the WCET?

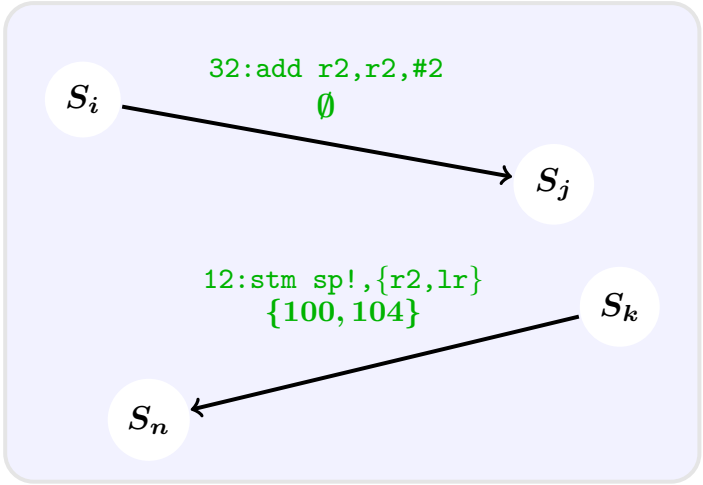
```
10:  e3a03000    mov   r3, #0
14:  e58d3014    str   r3, [sp, #20]
18:  e3a03002    mov   r3, #2
1c:  e58d300c    str   r3, [sp, #12]
20:  ea00000a    b     50 <fib+0x50>
24:  e59d3010    ldr   r3, [sp, #16]
28:  e58d3018    str   r3, [sp, #24]
2c:  e59d2010    ldr   r2, [sp, #16]
30:  e59d3014    ldr   r3, [sp, #20]
34:  e0823003    add   r3, r2, r3
38:  e58d3010    str   r3, [sp, #16]
3c:  e59d3018    ldr   r3, [sp, #24]
40:  e58d3014    str   r3, [sp, #20]
44:  e59d300c    ldr   r3, [sp, #12]
48:  e2833001    add   r3, r3, #1
4c:  e58d300c    str   r3, [sp, #12]
50:  e59d200c    ldr   r2, [sp, #12]
54:  e59d3004    ldr   r3, [sp, #4]
58:  e1520003    cmp   r2, r3
5c:  daffffb0    ble   24 <fib+0x24>
```



What do we Need to Compute the WCET?

```
10:  e3a03000    mov   r3, #0
14:  e58d3014    str   r3, [sp, #20]
18:  e3a03002    mov   r3, #2
1c:  e58d300c    str   r3, [sp, #12]
20:  ea00000a    b     50 <fib+0x50>
24:  e59d3010    ldr   r3, [sp, #16]
28:  e58d3018    str   r3, [sp, #24]
2c:  e59d2010    ldr   r2, [sp, #16]
30:  e59d3014    ldr   r3, [sp, #20]
34:  e0823003    add   r3, r2, r3
38:  e58d3010    str   r3, [sp, #16]
3c:  e59d3018    ldr   r3, [sp, #24]
40:  e58d3014    str   r3, [sp, #20]
44:  e59d300c    ldr   r3, [sp, #12]
48:  e2833001    add   r3, r3, #1
4c:  e58d300c    str   r3, [sp, #12]
50:  e59d200c    ldr   r2, [sp, #12]
54:  e59d3004    ldr   r3, [sp, #4]
58:  e1520003    cmp   r2, r3
5c:  dafffff0    ble   24 <fib+0x24>
```

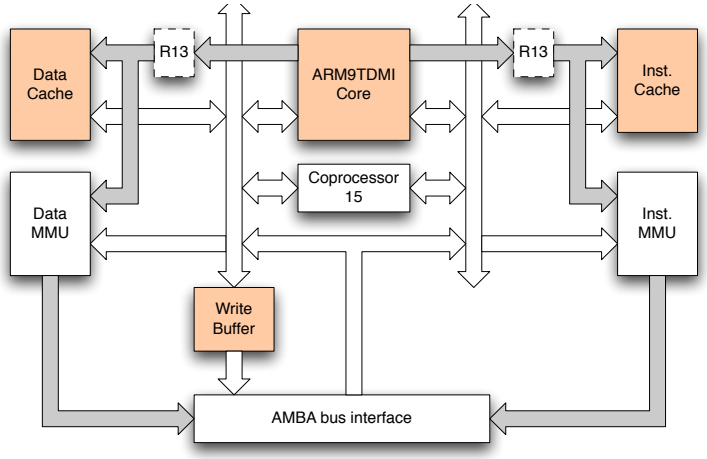


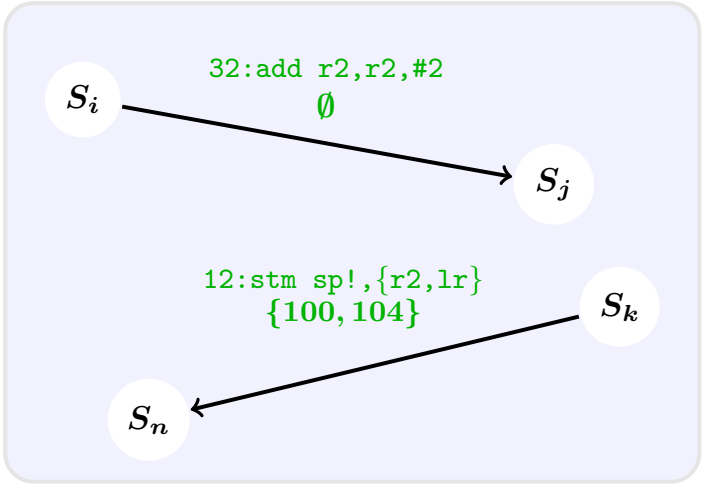


What do we Need to Compute the WCET?

```
10:  e3a03000    mov   r3, #0
14:  e58d3014    str   r3, [sp, #20]
18:  e3a03002    mov   r3, #2
1c:  e58d300c    str   r3, [sp, #12]
20:  ea00000a    b     50 <fib+0x50>
24:  e59d3010    ldr   r3, [sp, #16]
28:  e58d3018    str   r3, [sp, #24]
2c:  e59d2010    ldr   r2, [sp, #16]
30:  e59d3014    ldr   r3, [sp, #20]
34:  e0823003    add   r3, r2, r3
38:  e58d3010    str   r3, [sp, #16]
3c:  e59d3018    ldr   r3, [sp, #24]
40:  e58d3014    str   r3, [sp, #20]
44:  e59d300c    ldr   r3, [sp, #12]
48:  e2833001    add   r3, r3, #1
4c:  e58d300c    str   r3, [sp, #12]
50:  e59d200c    ldr   r2, [sp, #12]
54:  e59d3004    ldr   r3, [sp, #4]
58:  e1520003    cmp   r2, r3
5c:  dafffff0    ble   24 <fib+0x24>
```

$$\mathcal{L}(P) \subseteq \Sigma^*$$

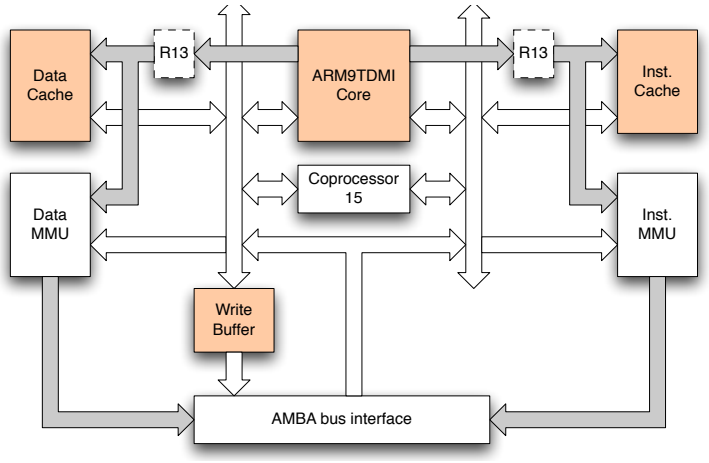




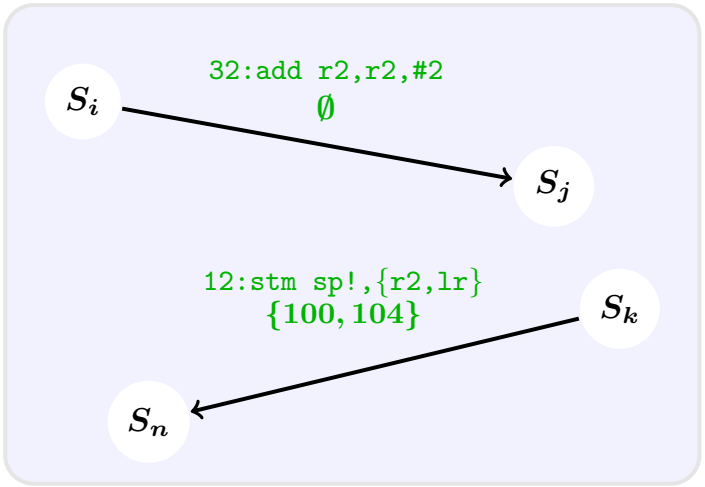
What do we Need to Compute the WCET?

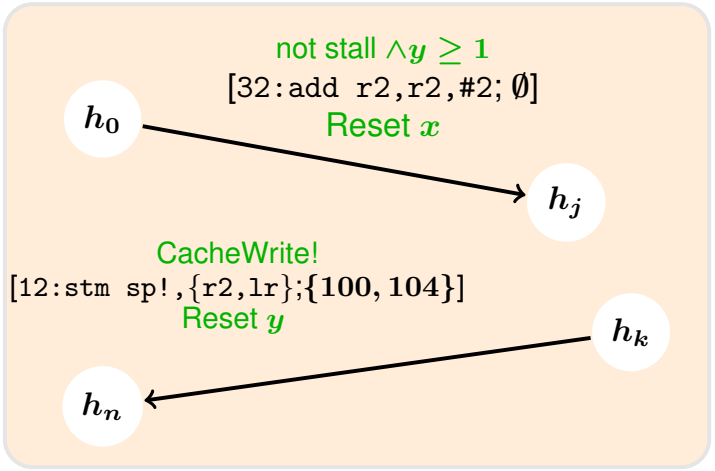
```
10:  e3a03000    mov   r3, #0
14:  e58d3014    str   r3, [sp, #20]
18:  e3a03002    mov   r3, #2
1c:  e58d300c    str   r3, [sp, #12]
20:  ea00000a    b     50 <fib+0x50>
24:  e59d3010    ldr   r3, [sp, #16]
28:  e58d3018    str   r3, [sp, #24]
2c:  e59d2010    ldr   r2, [sp, #16]
30:  e59d3014    ldr   r3, [sp, #20]
34:  e0823003    add   r3, r2, r3
38:  e58d3010    str   r3, [sp, #16]
3c:  e59d3018    ldr   r3, [sp, #24]
40:  e58d3014    str   r3, [sp, #20]
44:  e59d300c    ldr   r3, [sp, #12]
48:  e2833001    add   r3, r3, #1
4c:  e58d300c    str   r3, [sp, #12]
50:  e59d200c    ldr   r2, [sp, #12]
54:  e59d3004    ldr   r3, [sp, #4]
58:  e1520003    cmp   r2, r3
5c:  daffffff0   ble   24 <fib+0x24>
```

$\mathcal{L}(P) \subseteq \Sigma^*$



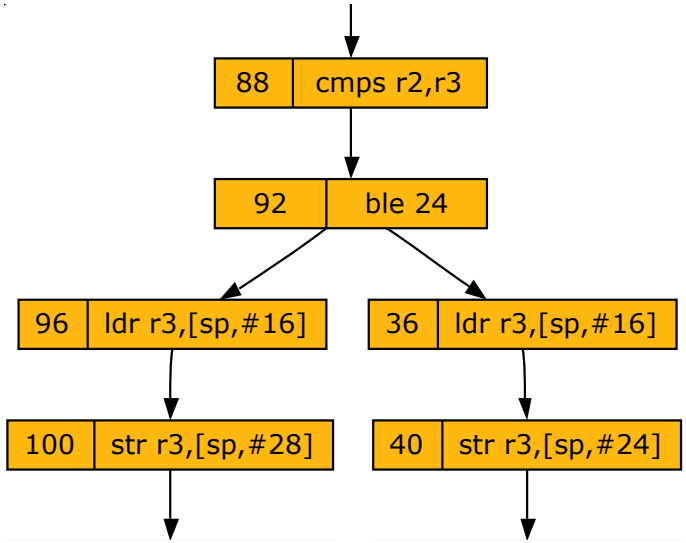
$\Sigma^* \longrightarrow \mathbb{N}$

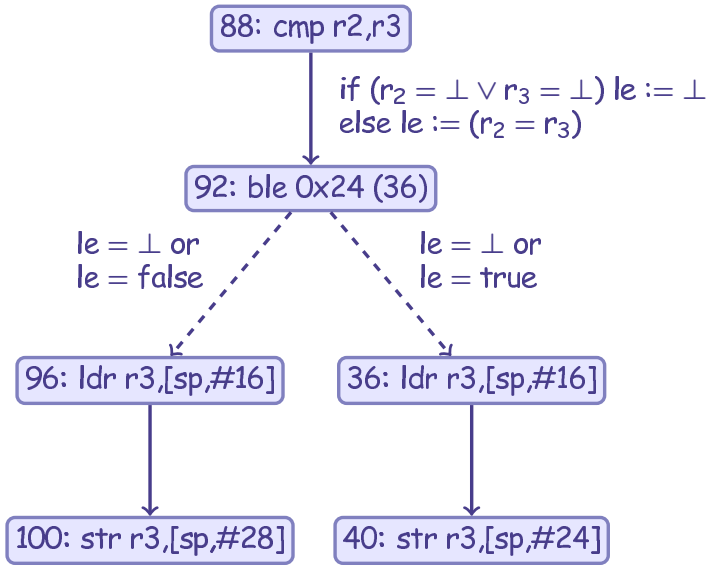




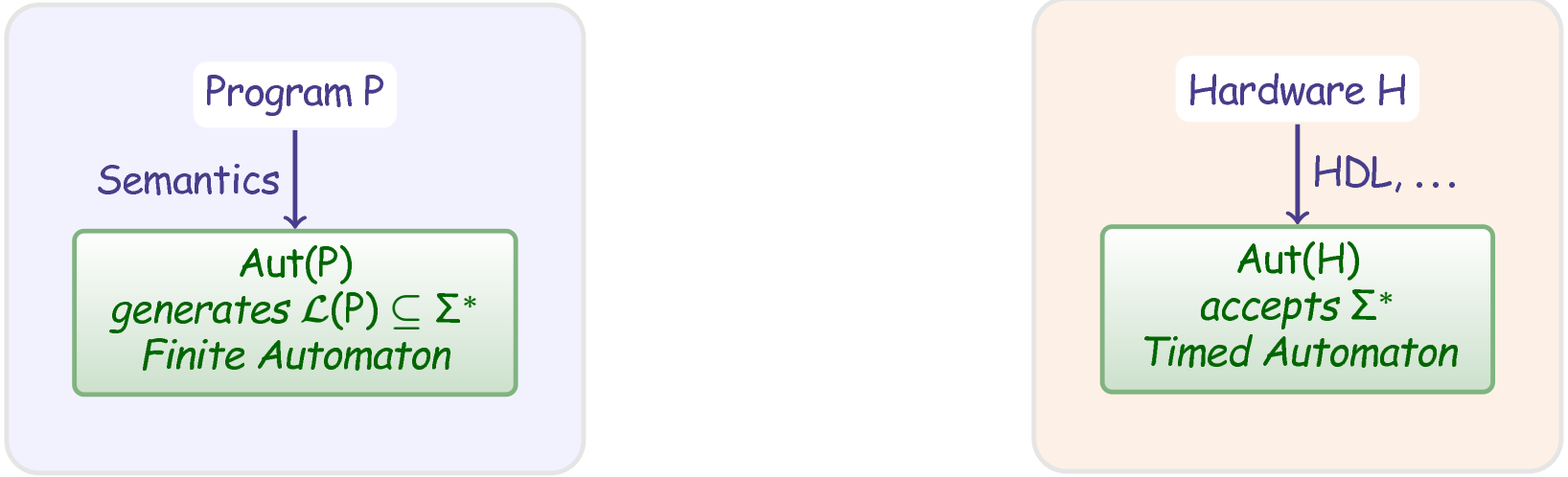
Non-Deterministic Choices

Input data are unknown: extended domain: $\mathcal{D}_\perp = \mathcal{D} \cup \{\perp\}$

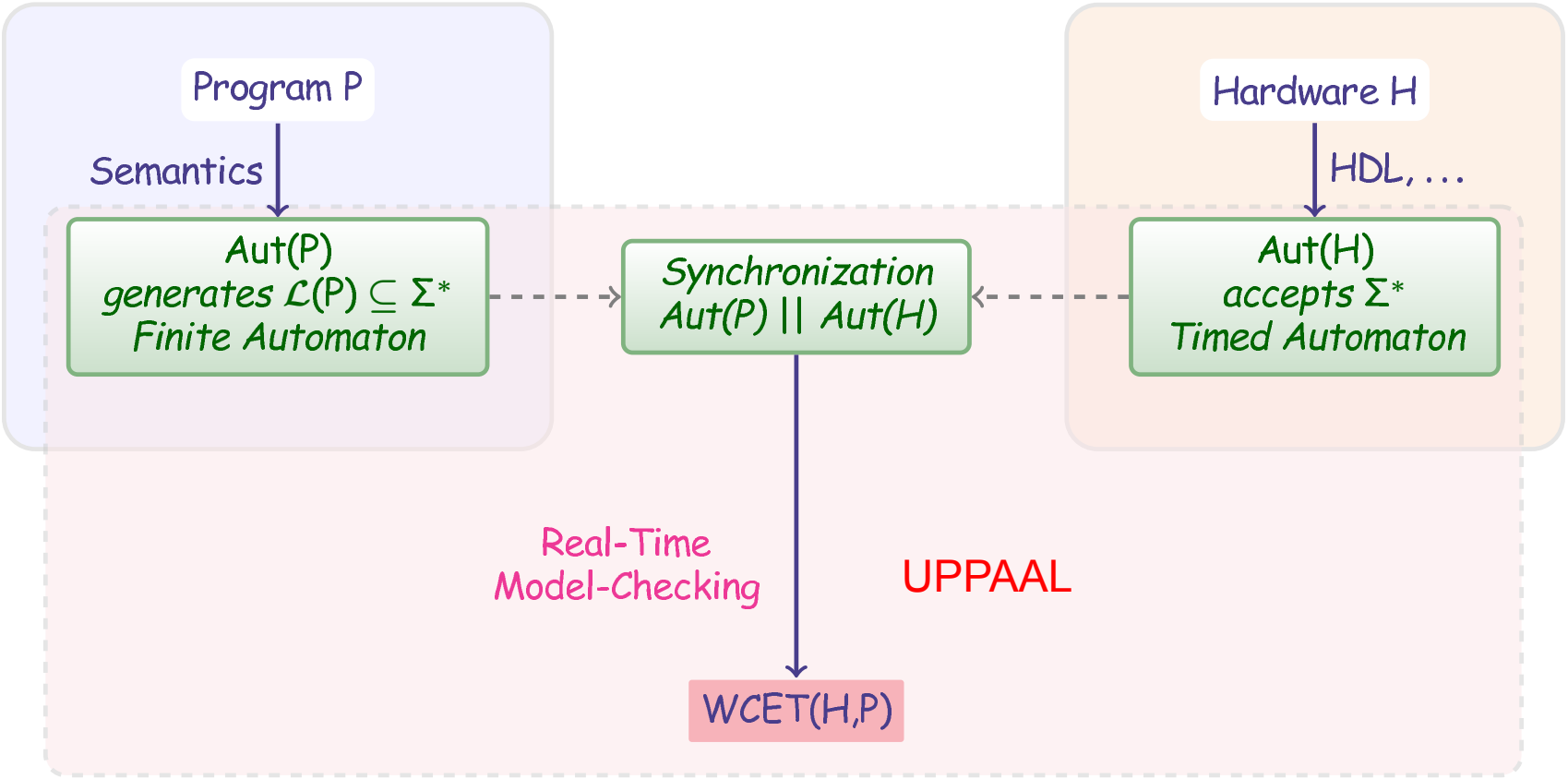




Modular Computation of WCET



Modular Computation of WCET



Reduced Automaton

- Two runs of P can generate the same word in $\mathcal{L}(P)$
  Fibonacci with initial values $u_0 = 0, u_1 = 1$ and $u_0 = 2, u_1 = 3$
- state of $Aut(P)$: 16 32-bit registers, stack, status bits

  size of a state of $Aut(P)$: $16 \times 32 + |stack| \times 32 + 4$

Reduced Automaton

- Two runs of P can generate the same word in $\mathcal{L}(P)$
  Fibonacci with initial values $u_0 = 0, u_1 = 1$ and $u_0 = 2, u_1 = 3$
- state of $Aut(P)$: 16 32-bit registers, stack, status bits

  size of a state of $Aut(P)$: $16 \times 32 + |stack| \times 32 + 4$
- WCET depends on $\mathcal{L}(P)$

  if $\mathcal{L}(P') = \mathcal{L}(P)$ then WCET$(H, P) =$ WCET(H, P')

Reduced Automaton

- Two runs of P can generate the same word in $\mathcal{L}(P)$
  Fibonacci with initial values $u_0 = 0, u_1 = 1$ and $u_0 = 2, u_1 = 3$
- state of $Aut(P)$: 16 32-bit registers, stack, status bits

  size of a state of $Aut(P)$: $16 \times 32 + |stack| \times 32 + 4$
- WCET depends on $\mathcal{L}(P)$

if $\mathcal{L}(P') = \mathcal{L}(P)$ then WCET$(H, P) =$ WCET(H, P')

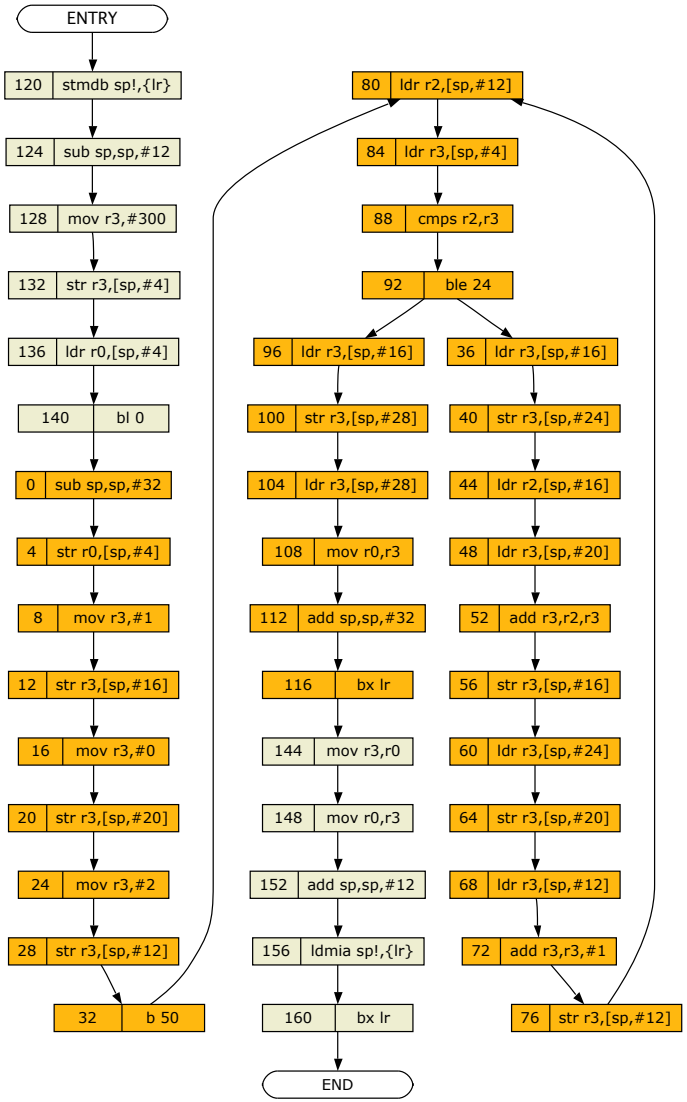
WCET-equivalent Program

P' and P are WCET-equivalent iff $\mathcal{L}(P') = \mathcal{L}(P)$.

Compute a reduced WCET-equivalent P' using Program Slicing

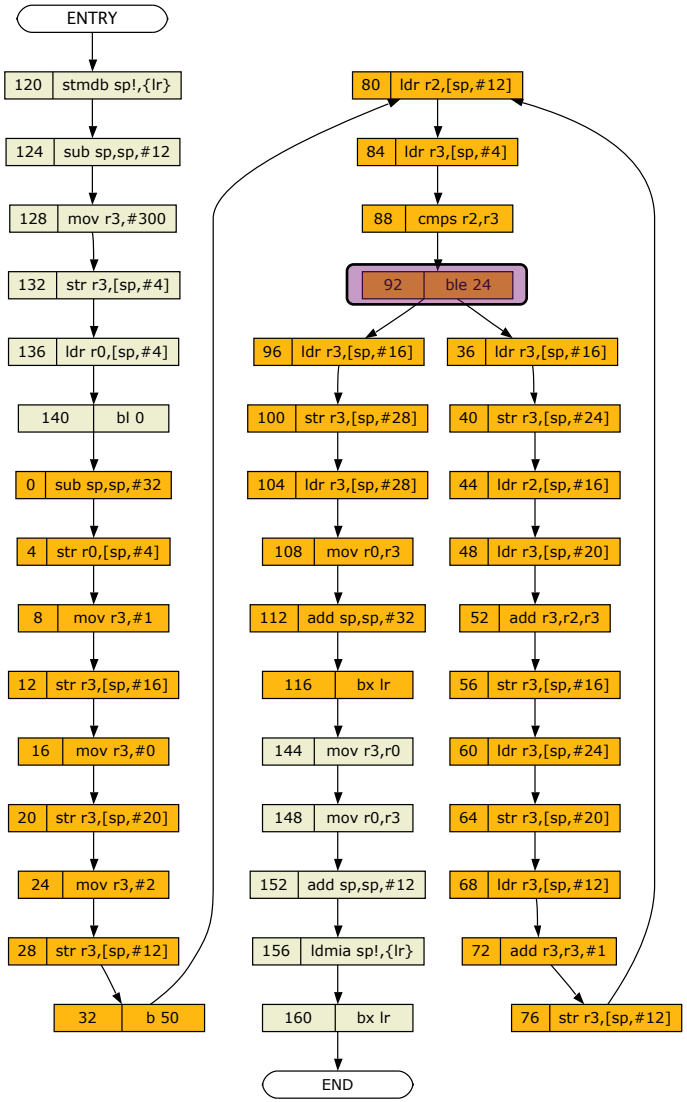
WCET-Equivalent Fibonnacci

```
00000000 <fib>:
   0:  e24dd020     sub   sp, sp, #32
   4:  e58d0004     str   r0, [sp, #4]
   8:  e3a03001     mov   r3, #1
   c:  e58d3010     str   r3, [sp, #16]
  10:  e3a03000     mov   r3, #0
  14:  e58d3014     str   r3, [sp, #20]
  18:  e3a03002     mov   r3, #2
  1c:  e58d300c     str   r3, [sp, #12]
  20:  ea00000a     b     50 <fib+0x50>
  24:  e59d3010     ldr   r3, [sp, #16]
  28:  e58d3018     str   r3, [sp, #24]
  2c:  e59d2010     ldr   r2, [sp, #16]
  30:  e59d3014     ldr   r3, [sp, #20]
  34:  e0823003     add   r3, r2, r3
  38:  e58d3010     str   r3, [sp, #16]
  3c:  e59d3018     ldr   r3, [sp, #24]
  40:  e58d3014     str   r3, [sp, #20]
  44:  e59d300c     ldr   r3, [sp, #12]
  48:  e2833001     add   r3, r3, #1
  4c:  e58d300c     str   r3, [sp, #12]
  50:  e59d200c     ldr   r2, [sp, #12]
  54:  e59d3004     ldr   r3, [sp, #4]
  58:  e1520003     cmp   r2, r3
  5c:  dafffff0     ble   24 <fib+0x24>
  60:  e59d3010     ldr   r3, [sp, #16]
  64:  e58d301c     str   r3, [sp, #28]
  68:  e59d301c     ldr   r3, [sp, #28]
  6c:  e1a00003     mov   r0, r3
  70:  e28dd020     add   sp, sp, #32
  74:  e12fff1e     bx    lr

00000078 <main>:
  78:  e52de004     push  {lr}
  7c:  e24dd00c     sub   sp, sp, #12
  80:  e3a03f4b     mov   r3, #300
  84:  e58d3004     str   r3, [sp, #4]
  88:  e59d0004     ldr   r0, [sp, #4]
  8c:  ebfffffb     bl    0 <fib>
  90:  e1a03000     mov   r3, r0
  94:  e1a00003     mov   r0, r3
  98:  e28dd00c     add   sp, sp, #12
  9c:  e49de004     pop   {lr}
  a0:  e12fff1e     bx    lr
```



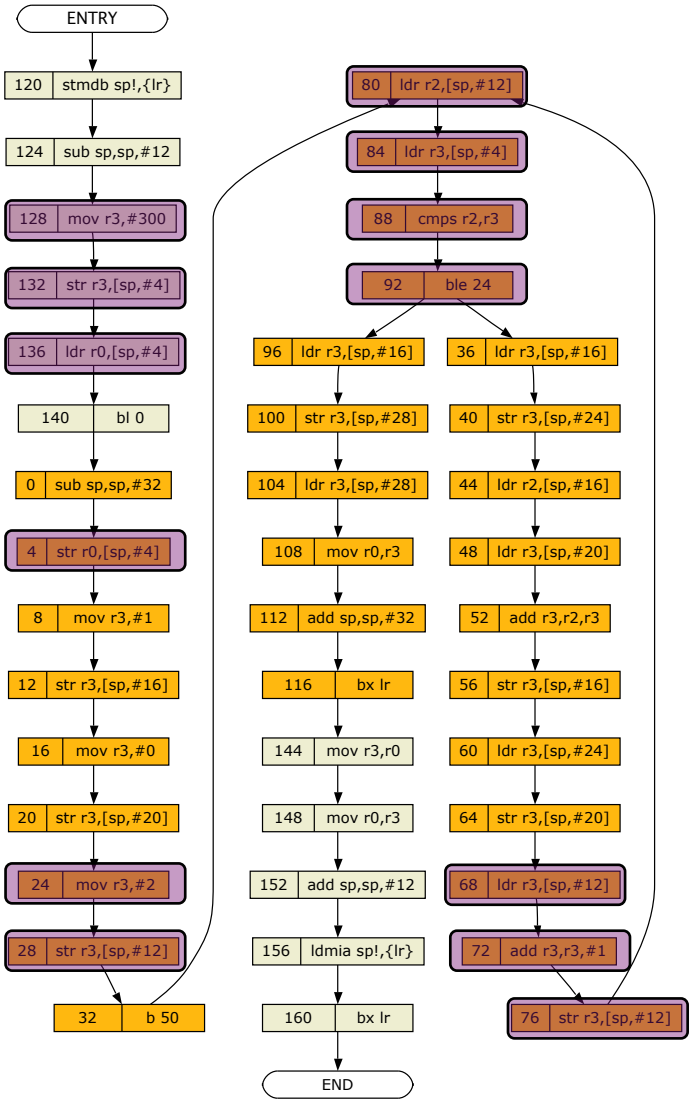
WCET-Equivalent Fibonnacci

```
00000000 <fib>:
   0:  e24dd020     sub   sp, sp, #32
   4:  e58d0004     str   r0, [sp, #4]
   8:  e3a03001     mov   r3, #1
   c:  e58d3010     str   r3, [sp, #16]
  10:  e3a03000     mov   r3, #0
  14:  e58d3014     str   r3, [sp, #20]
  18:  e3a03002     mov   r3, #2
  1c:  e58d300c     str   r3, [sp, #12]
  20:  ea00000a     b     50 <fib+0x50>
  24:  e59d3010     ldr   r3, [sp, #16]
  28:  e58d3018     str   r3, [sp, #24]
  2c:  e59d2010     ldr   r2, [sp, #16]
  30:  e59d3014     ldr   r3, [sp, #20]
  34:  e0823003     add   r3, r2, r3
  38:  e58d3010     str   r3, [sp, #16]
  3c:  e59d3018     ldr   r3, [sp, #24]
  40:  e58d3014     str   r3, [sp, #20]
  44:  e59d300c     ldr   r3, [sp, #12]
  48:  e2833001     add   r3, r3, #1
  4c:  e58d300c     str   r3, [sp, #12]
  50:  e59d200c     ldr   r2, [sp, #12]
  54:  e59d3004     ldr   r3, [sp, #4]
  58:  e1520003     cmp   r2, r3
  5c:  dafffff0     ble   24 <fib+0x24>
  60:  e59d3010     ldr   r3, [sp, #16]
  64:  e58d301c     str   r3, [sp, #28]
  68:  e59d301c     ldr   r3, [sp, #28]
  6c:  e1a00003     mov   r0, r3
  70:  e28dd020     add   sp, sp, #32
  74:  e12fff1e     bx    lr

00000078 <main>:
  78:  e52de004     push  {lr}
  7c:  e24dd00c     sub   sp, sp, #12
  80:  e3a03f4b     mov   r3, #300
  84:  e58d3004     str   r3, [sp, #4]
  88:  e59d0004     ldr   r0, [sp, #4]
  8c:  ebfffffb     bl    0 <fib>
  90:  e1a03000     mov   r3, r0
  94:  e1a00003     mov   r0, r3
  98:  e28dd00c     add   sp, sp, #12
  9c:  e49de004     pop   {lr}
  a0:  e12fff1e     bx    lr
```



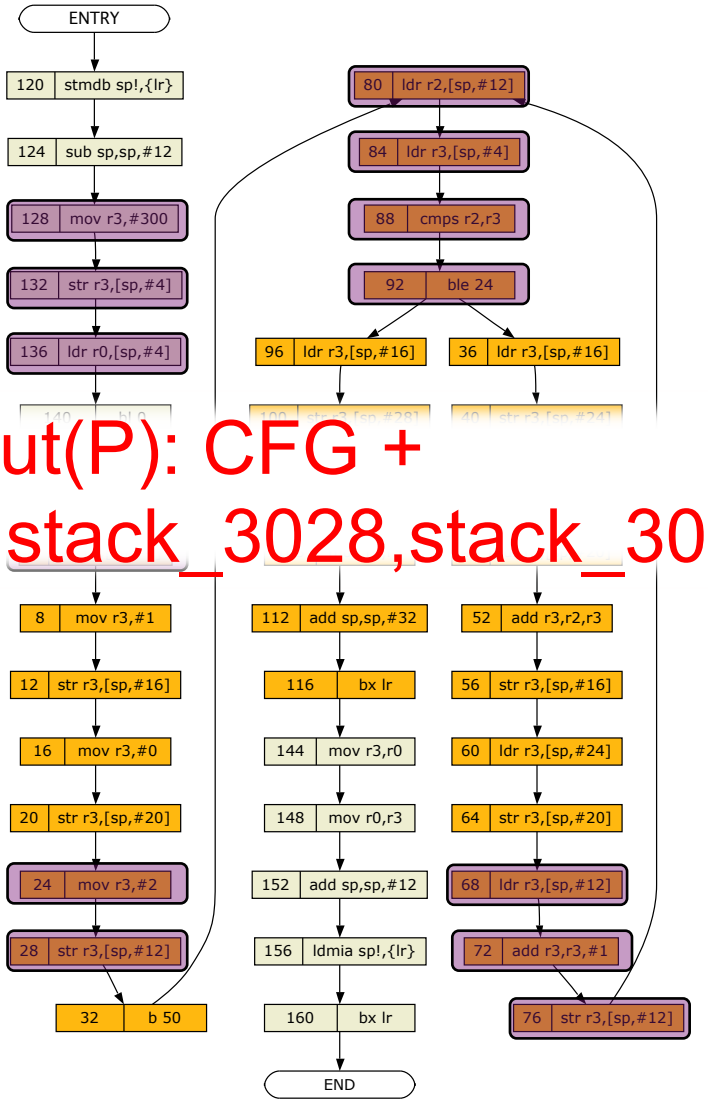
WCET-Equivalent Fibonnacci

```
00000000 <fib>:
  0:  e24dd020    sub   sp, sp, #32
  4:  e58d0004    str   r0, [sp, #4]
  8:  e3a03001    mov   r3, #1
  c:  e58d3010    str   r3, [sp, #16]
 10:  e3a03000    mov   r3, #0
 14:  e58d3014    str   r3, [sp, #20]
 18:  e3a03002    mov   r3, #2
 1c:  e58d300c    str   r3, [sp, #12]
 20:  ea00000a    b     50 <fib+0x50>
 24:  e59d3010    ldr   r3, [sp, #16]
 28:  e58d3018    str   r3, [sp, #24]
 2c:  e59d2010    ldr   r2, [sp, #16]
 30:  e59d3014    ldr   r3, [sp, #20]
 34:  e0823003    add   r3, r2, r3
 38:  e58d3010    str   r3, [sp, #16]
 3c:  e59d3018    ldr   r3, [sp, #24]
 40:  e58d3014    str   r3, [sp, #20]
 44:  e59d300c    ldr   r3, [sp, #12]
 48:  e2833001    add   r3, r3, #1
 4c:  e58d300c    str   r3, [sp, #12]
 50:  e59d200c    ldr   r2, [sp, #12]
 54:  e59d3004    ldr   r3, [sp, #4]
 58:  e1520003    cmp   r2, r3
 5c:  dafffff0    ble   24 <fib+0x24>
 60:  e59d3010    ldr   r3, [sp, #16]
 64:  e58d301c    str   r3, [sp, #28]
 68:  e59d301c    ldr   r3, [sp, #28]
 6c:  e1a00003    mov   r0, r3
 70:  e28dd020    add   sp, sp, #32
 74:  e12fff1e    bx    lr

00000078 <main>:
 78:  e52de004    push  {lr}
 7c:  e24dd00c    sub   sp, sp, #12
 80:  e3a03f4b    mov   r3, #300
 84:  e58d3004    str   r3, [sp, #4]
 88:  e59d0004    ldr   r0, [sp, #4]
 8c:  ebffffdb    bl    0 <fib>
 90:  e1a03000    mov   r3, r0
 94:  e1a00003    mov   r0, r3
 98:  e28dd00c    add   sp, sp, #12
 9c:  e49de004    pop   {lr}
 a0:  e12fff1e    bx    lr
```



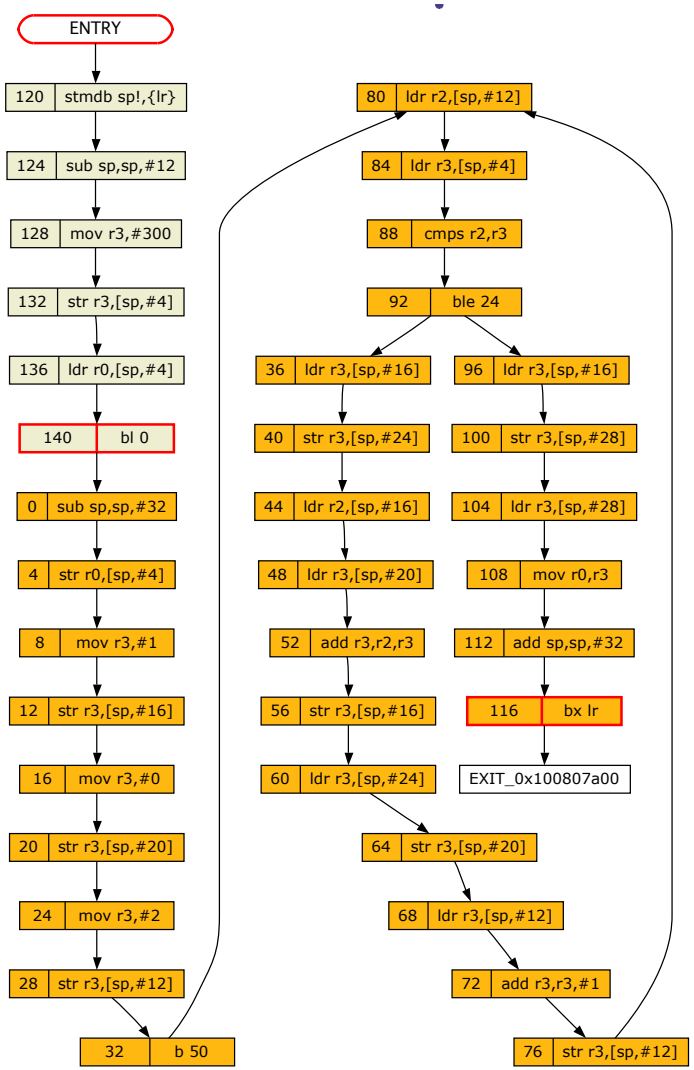
WCET-Equivalent Fibonnacci

```
00000000 <fib>:
  0:  e24dd020    sub   sp, sp, #32
  4:  e58d0004    str   r0, [sp, #4]
  8:  e3a03001    mov   r3, #1
  c:  e58d3010    str   r3, [sp, #16]
 10:  e3a03000    mov   r3, #0
 14:  e58d3014    str   r3, [sp, #20]
 18:  e3a03002    mov   r3, #2
 1c:  e58d300c    str   r3, [sp, #12]
 20:  ea00000a    b     50 <fib+0x50>
 24:  e59d3010    ldr   r3, [sp, #16]
 28:  e58d3018    str   r3, [sp, #24]
 2c:  e59d2010    ldr   r2, [sp, #16]
 30:  e59d3014    ldr   r3, [sp, #20]
 34:  e0823003    add   r3, r2, r3
 38:  e58d3010    str   r3, [sp, #16]
 3c:  e59d3018    ldr   r3, [sp, #24]
 40:  e58d301
 44:  e59d30
 48:  e2833
 4c:  e58d30
 50:  e59d20
 54:  e59d30
 58:  e1520003    cmp   r2, r3
 5c:  dafffff0    ble   24 <fib+0x24>
 60:  e59d3010    ldr   r3, [sp, #16]
 64:  e58d301c    str   r3, [sp, #28]
 68:  e59d301c    ldr   r3, [sp, #28]
 6c:  e1a00003    mov   r0, r3
 70:  e28dd020    add   sp, sp, #32
 74:  e12fff1e    bx    lr

00000078 <main>:
 78:  e52de004    push  {lr}
 7c:  e24dd00c    sub   sp, sp, #12
 80:  e3a03f4b    mov   r3, #300
 84:  e58d3004    str   r3, [sp, #4]
 88:  e59d0004    ldr   r0, [sp, #4]
 8c:  ebfffffb    bl    0 <fib>
 90:  e1a03000    mov   r3, r0
 94:  e1a00003    mov   r0, r3
 98:  e28dd00c    add   sp, sp, #12
 9c:  e49de004    pop   {lr}
 a0:  e12fff1e    bx    lr
```

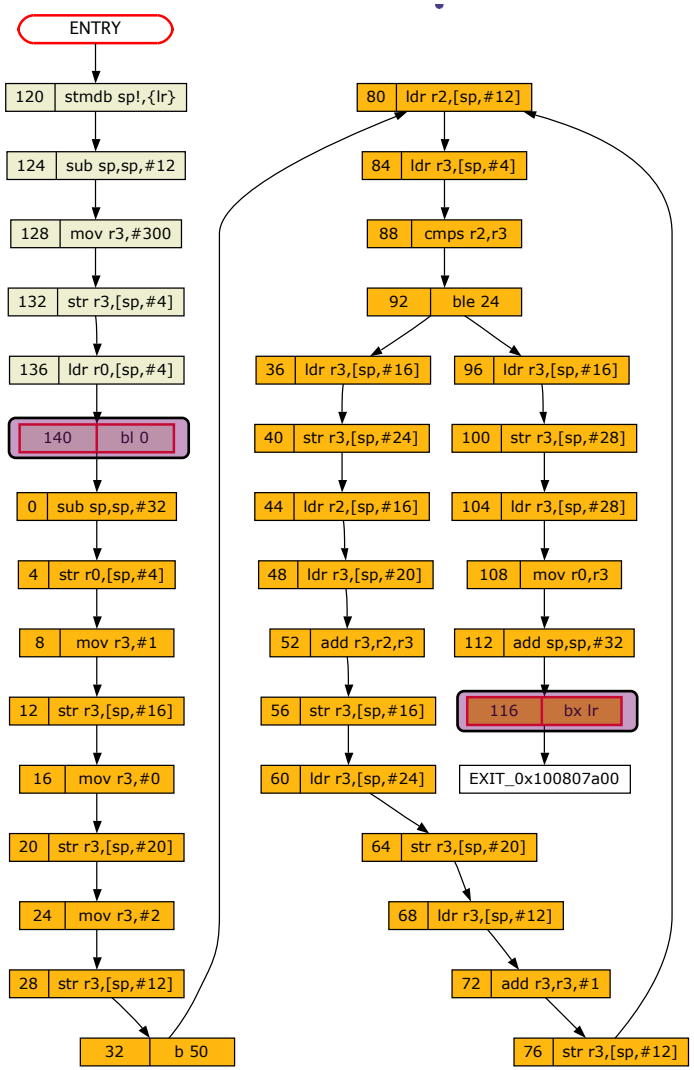


Size of Aut(P): CFG + r0,r2,r3,stack_3020,stack_3028,stack_3052

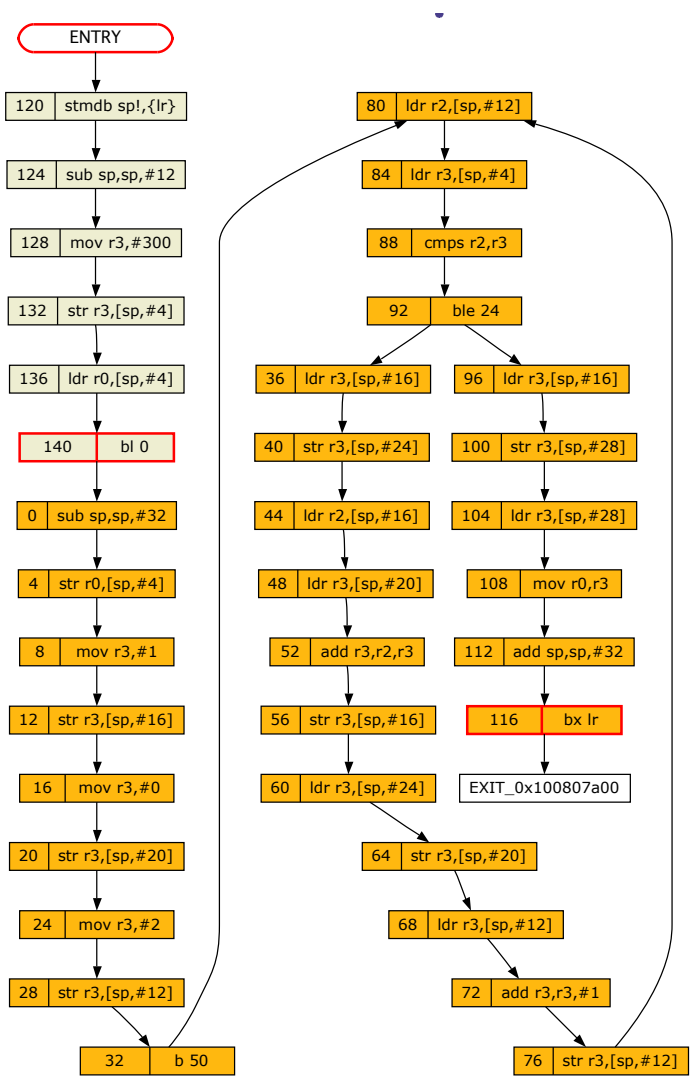
Slicing ... Can also be Used to Build CFG

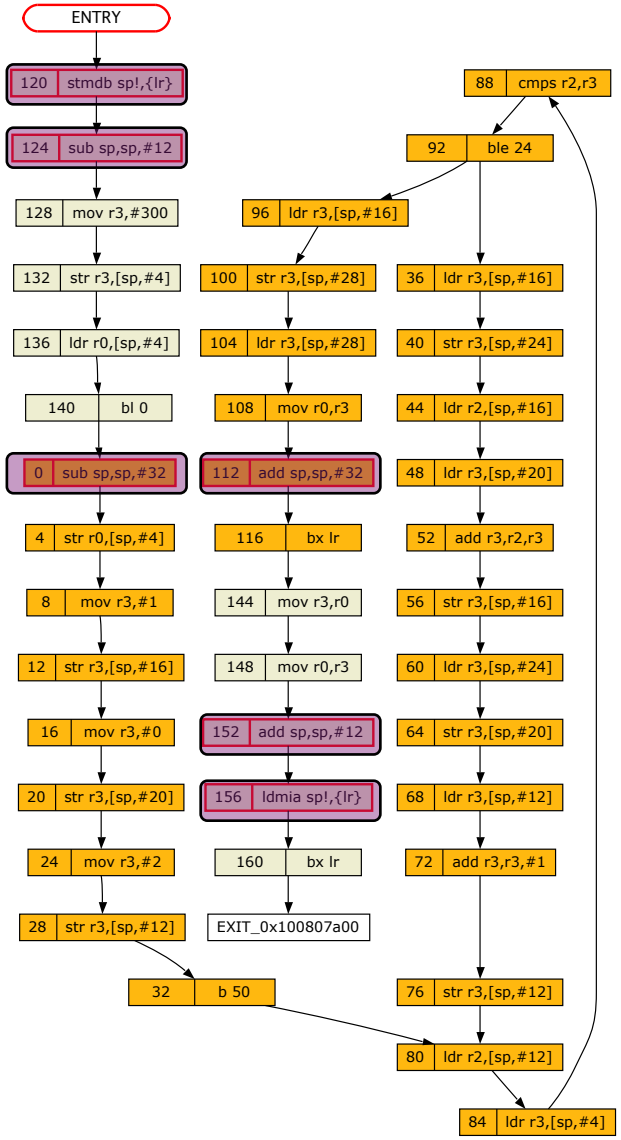


Slicing ... Can also be Used to Build CFG

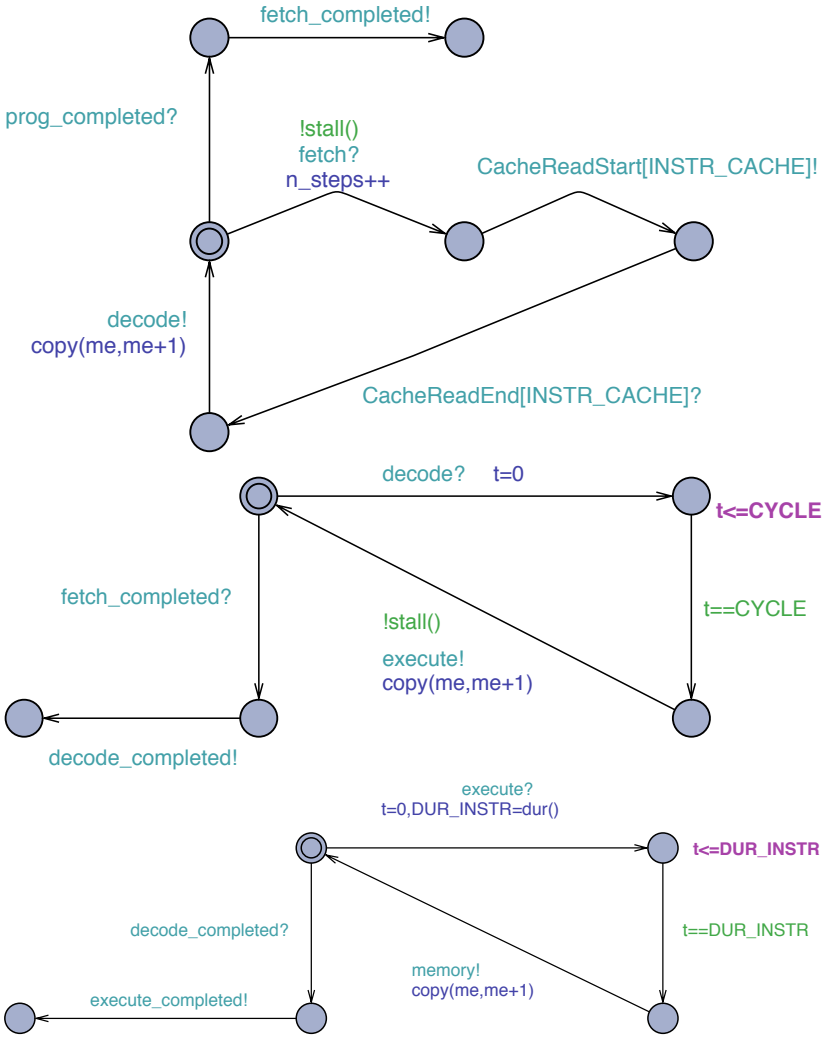


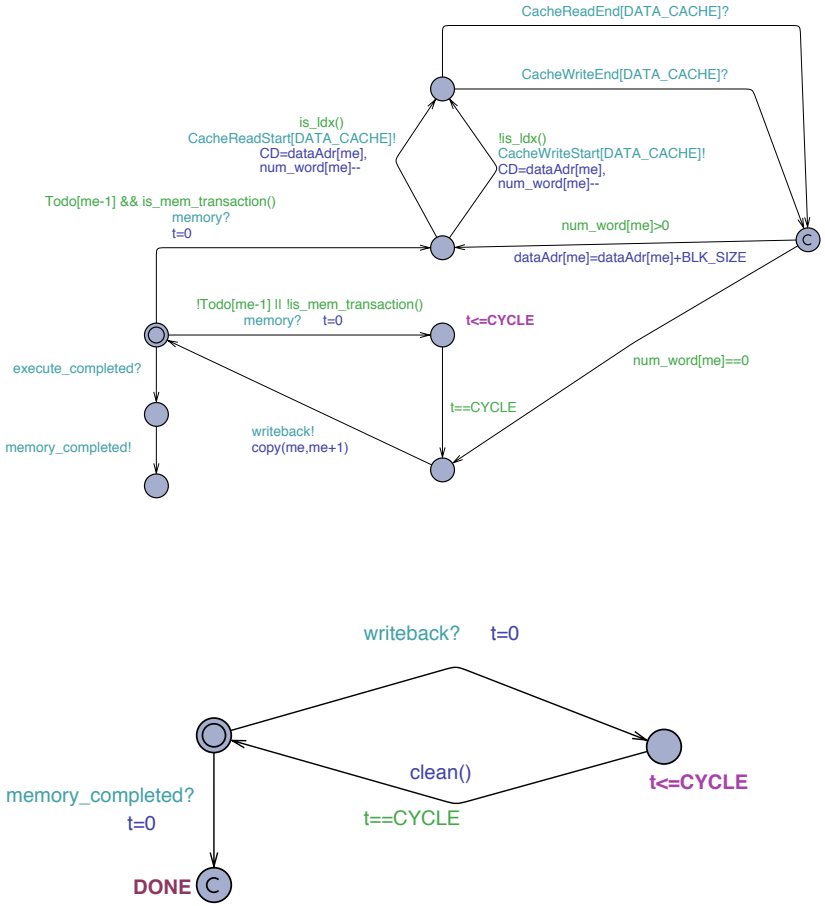
Slicing ... Can also be Used to Build CFG



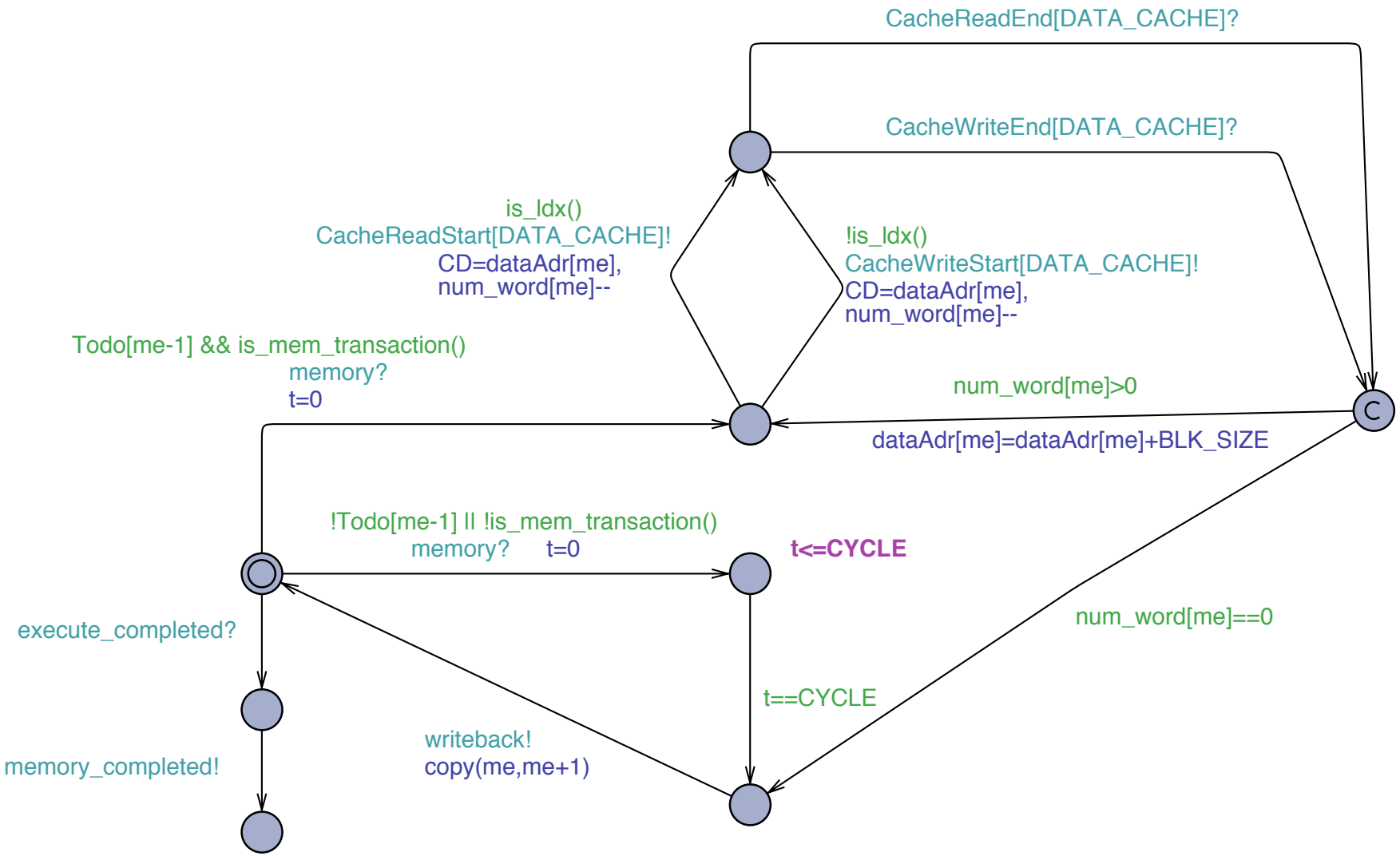


Pipeline (except Memory stage)

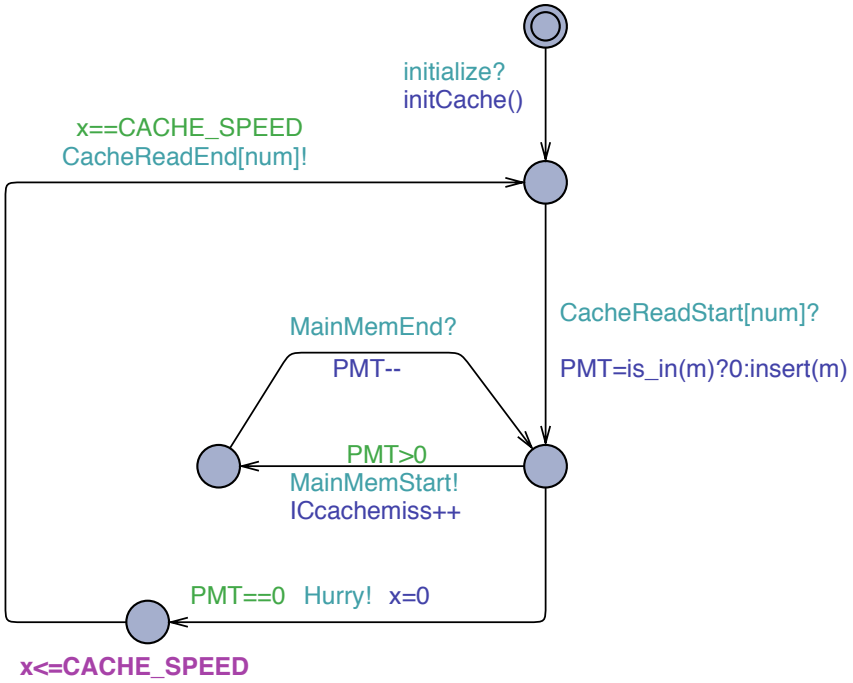


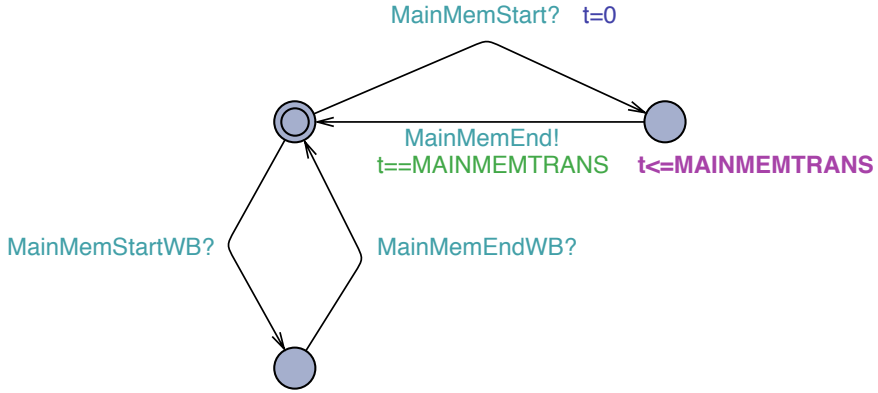


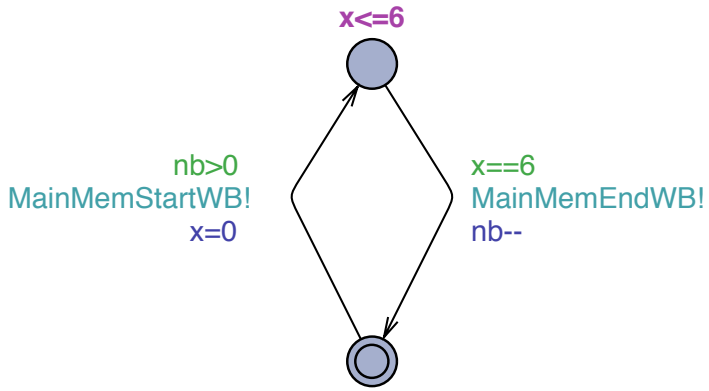
Hardware Formal Models: Memory Stage of Pipeline



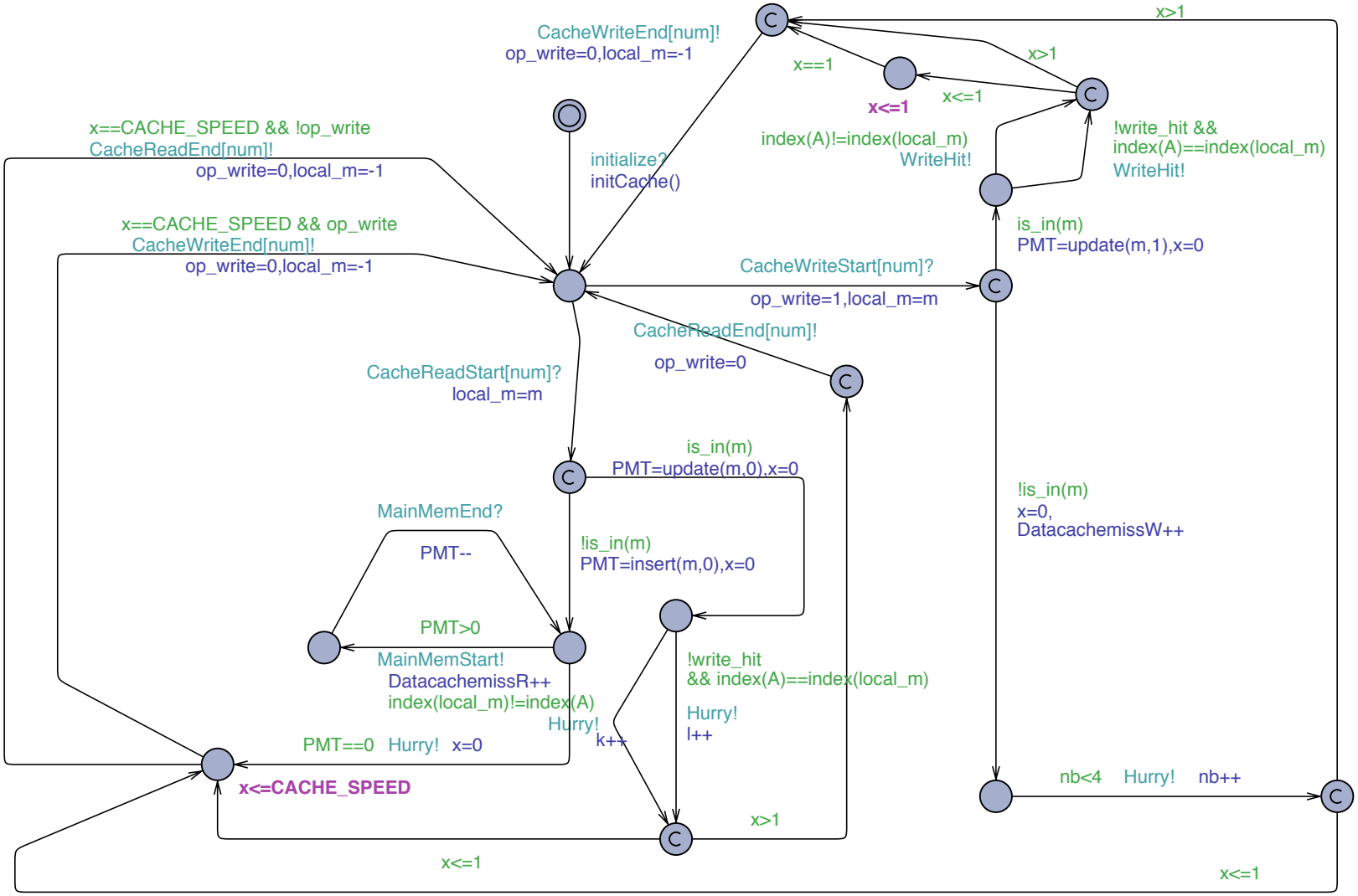
Caches and Main Memory



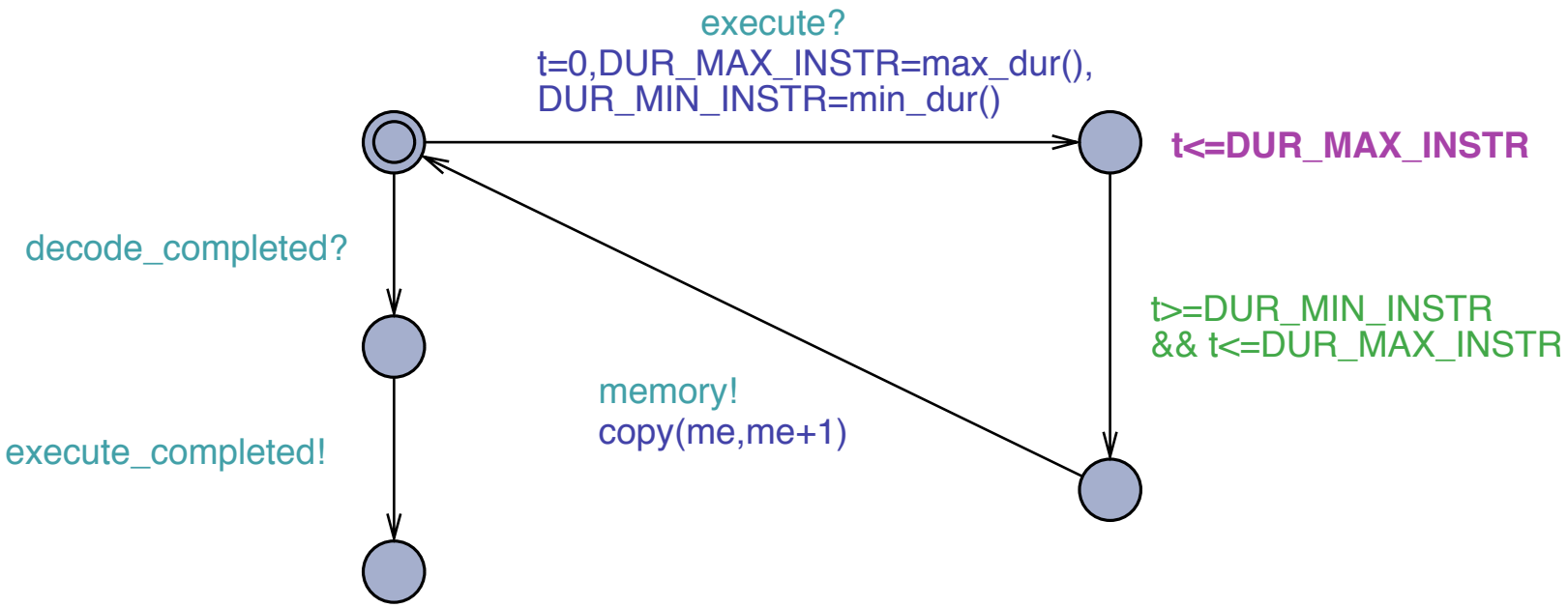




Data Cache



Data Dependent Timing



Measuring Execution Times

```
#define timerToCPUClockRatio 12

main ()
{
  int result;
  unsigned int start;
  unsigned int stop;

  start = timerGetValue(1);
  result = fib(300);
  stop = timerGetValue(1);
  printf("fib(300):␣%d,␣time=%lu\n", result,
                (stop-start)*timerToCPUClockRatio);
  while (1);
}
```

- Embedded hardware timer: $1/12$th of processor clock frequency
- measurement error is ± 24 cycles
- a program executing in $\geq$ 1200 cycles may be accurately measured
  less than 1% of measurement error

Experimental Results

Program⊕	loc†	UPPAAL Time/States Explored¶	Computed BCET/WCET (C)	Measured BCET/WCET (M)	Error (%)‡	Slice§
Single-Path Programs						
fib-O0	74	2s/74181	8098	8064	**0.42%**	47/131
fib-O1	74	0.6s/22333	2597	2544	**2.0%**	18/72
fib-O2	74	0.3s/9711	1209	1164	**3.8%**	22/71
janne-complex-O0*	65	1.7s/38038	4264	4164	**2.4%**	78/173
janne-complex-O1*	65	0.5s/14600	1715	1680	**2.0%**	30/89
janne-complex-O2*	65	0.5s/13004	1557	1536	**1.3%**	32/78
fdct-O1	238	21s/60534	4245	4092	**3.7%**	100/363
fdct-O2	238	3.24s/55285	19231	18984	**1.3%**	166/3543
Single-Path Programs‡ with MUL/MLA/SMULL instructions (duration of instruction depends on data)						
fdct-O0	238	124s/85008	11242/11800	11448	**3.0%**	253/831
matmult-O0*	162	217s/10531262	502849/529250	511584/528684	**0.1%**	158/314
matmult-O1*	162	25s/1112527	129967/156367	127356/153000	**2.2%**	71/172
matmult-O2*	162	121s/6780931	122045/148299	116844/140664	**5.4%**	75/288
jfdcint-O0	374	92s/100861	12726/12918	12588	**2.6%**	159/792
jfdcint-O1	374	12s/35419	4880/5072	4668	**8.6%**	25/325
jfdcint-O2	374	5.38s/175661	[16746,16938]	16380	**3.4%**	56/2512
Multiple-Path Programs						
bs-O0	174	30s/1421274	478/1068	1056	**1.1%**	75/151
bs-O1	174	23s/1214673	321/738	720	**2.5%**	28/82
bs-O2	174	12s/655870	273/628	600	**4.6%**	28/65
cnt-O0*	115	4s/77002	9025/9027	8836	**2.1%**	99/235
cnt-O1*	115	1.4s/27146	4123/4123	3996	**3.1%**	42/129
cnt-O2*	115	9s/11490	3067/3067	2928	**4.6%**	39/263
insertsort-O0*	91	598.98s/24250738	3133	3108	**0.8%**	79/175
insertsort-O1*	91	353.80s/11455293	1533	1500	**2.2%**	40/115
insertsort-O2*	91	11.68s/387292	1326	1320	**0.4%**	43/108
ns-O0*	497	60s/3064316	940/30968	30732	**0.8%**	132/215
ns-O1*	497	8s/368720	605/11701	11568	**1.1%**	61/124
ns-O2*	497	55s/1030746	441/7280	7236	**0.6%**	566/863

⊕ file-Ox indicates that file was compiled using gcc -Ox

† lines of code in the C source file

‡ $\frac{(C-M)}{M} \times 100$ computed using the upper bound for C and M

§ Instructions in Slice/Instructions in Program

* Program selected for the WCET Challenge 2006

¶ UPPAAL 4.1.11/Intel Pentium 5/3.1Ghz/16GB

Experimental Results

NICTA

Program⊕	loc†	UPPAAL Time/States Explored¶	Computed BCET/WCET (C)	Measured BCET/WCET (M)	Error (%)‡	Slice§
Single-Path Programs						
fib-O0	74	2s/74181	8098	8064	**0.42%**	47/131
fib-O1	74	0.6s/22333	2597	2544	**2.0%**	18/72
fib-O2	74	0.3s/9711	1209	1164	**3.8%**	22/71
janne-complex-O0*	65	1.7s/38038	4264	4164	**2.4%**	78/173
janne-complex-O1*	65	0.5s/14600	1715	1680	**2.0%**	30/89
janne-complex-O2*	65	0.5s/13004	1557	1536	**1.3%**	32/78
fdct-O1	238	21s/60534	4245	4092	**3.7%**	100/363
fdct-O2	238	3.24s/55285	19231	18984	**1.3%**	166/3543
Single-Path Programs‡ with MUL/MLA/SMULL instructions (duration of instruction depends on data)						
fdct-O0	238	124s/85008	11242/11800	11448	**3.0%**	253/831
matmult-O0*	162	217s/10531262	502849/529250	511584/528684	**0.1%**	158/314
matmult-O1*	162	25s/1112527	129967/156367	127356/153000	**2.2%**	71/172
matmult-O2*	162	121s/6780931	122045/148299	116844/140664	**5.4%**	75/288
jfdcint-O0	374	92s/100861	12726/12918	12588	**2.6%**	159/792
jfdcint-O1	374	12s/35419	4880/5072	4668	**8.6%**	25/325
jfdcint-O2	374	5.38s/175661	[16746,16938]	16380	**3.4%**	56/2512
Multiple-Path Programs						
bs-O0	174	30s/1421274	478/1068	1056	**1.1%**	75/151
bs-O1	174	23s/1214673	321/738	720	**2.5%**	28/82
bs-O2	174	12s/655870	273/628	600	**4.6%**	28/65
cnt-O0*	115	4s/77002	9025/9027	8836	**2.1%**	99/235
cnt-O1*	115	1.4s/27146	4123/4123	3996	**3.1%**	42/129
cnt-O2*	115	9s/11490	3067/3067	2928	**4.6%**	39/263
insertsort-O0*	91	598.98s/24250738	3133	3108	**0.8%**	79/175
insertsort-O1*	91	353.80s/11455293	1533	1500	**2.2%**	40/115
insertsort-O2*	91	11.68s/387292	1326	1320	**0.4%**	43/108
ns-O0*	497	60s/3064316	940/30968	30732	**0.8%**	132/215
ns-O1*	497	8s/368720	605/11701	11568	**1.1%**	61/124
ns-O2*	497	55s/1030746	441/7280	7236	**0.6%**	566/863

⊕ file-Ox indicates that file was compiled using gcc -Ox

† lines of code in the C source file

‡ $\frac{(C-M)}{M} \times 100$ computed using the upper bound for C and M

§ Instructions in Slice/Instructions in Program

* Program selected for the WCET Challenge 2006

¶ UPPAAL 4.1.11/Intel Pentium 5/3.1Ghz/16GB

Experimental Results

NICTA

Program⊕	loc†	UPPAAL Time/States Explored¶	Computed BCET/WCET (C)	Measured BCET/WCET (M)	Error (%)‡	Slice§
Single-Path Programs						
fib-O0	74	2s/74181	8098	8064	**0.42%**	47/131
fib-O1	74	0.6s/22333	2597	2544	**2.0%**	18/72
fib-O2	74	0.3s/9711	1209	1164	**3.8%**	22/71
janne-complex-O0*	65	1.7s/38038	4264	4164	**2.4%**	78/173
janne-complex-O1*	65	0.5s/14600	1715	1680	**2.0%**	30/89
janne-complex-O2*	65	0.5s/13004	1557	1536	**1.3%**	32/78
fdct-O1	238	21s/60534	4245	4092	**3.7%**	100/363
fdct-O2	238	3.24s/55285	19231	18984	**1.3%**	166/3543
Single-Path Programs‡ with MUL/MLA/SMULL instructions (duration of instruction depends on data)						
fdct-O0	238	124s/85008	11242/11800	11448	**3.0%**	253/831
matmult-O0*	162	217s/10531262	502849/529250	511584/528684	**0.1%**	158/314
matmult-O1*	162	25s/1112527	129967/156367	127356/153000	**2.2%**	71/172
matmult-O2*	162	121s/6780931	122045/148299	116844/140664	**5.4%**	75/288
jfdcint-O0	374	92s/100861	12726/12918	12588	**2.6%**	159/792
jfdcint-O1	374	12s/35419	4880/5072	4668	**8.6%**	25/325
jfdcint-O2	374	5.38s/175661	[16746,16938]	16380	**3.4%**	56/2512
Multiple-Path Programs						
bs-O0	174	30s/1421274	478/1068	1056	**1.1%**	75/151
bs-O1	174	23s/1214673	321/738	720	**2.5%**	28/82
bs-O2	174	12s/655870	273/628	600	**4.6%**	28/65
cnt-O0*	115	4s/77002	9025/9027	8836	**2.1%**	99/235
cnt-O1*	115	1.4s/27146	4123/4123	3996	**3.1%**	42/129
cnt-O2*	115	9s/11490	3067/3067	2928	**4.6%**	39/263
insertsort-O0*	91	598.98s/24250738	3133	3108	**0.8%**	79/175
insertsort-O1*	91	353.80s/11455293	1533	1500	**2.2%**	40/115
insertsort-O2*	91	11.68s/387292	1326	1320	**0.4%**	43/108
ns-O0*	497	60s/3064316	940/30968	30732	**0.8%**	132/215
ns-O1*	497	8s/368720	605/11701	11568	**1.1%**	61/124
ns-O2*	497	55s/1030746	441/7280	7236	**0.6%**	566/863

⊕ file-Ox indicates that file was compiled using gcc -Ox

† lines of code in the C source file

‡ $\frac{(C-M)}{M} \times 100$ computed using the upper bound for C and M

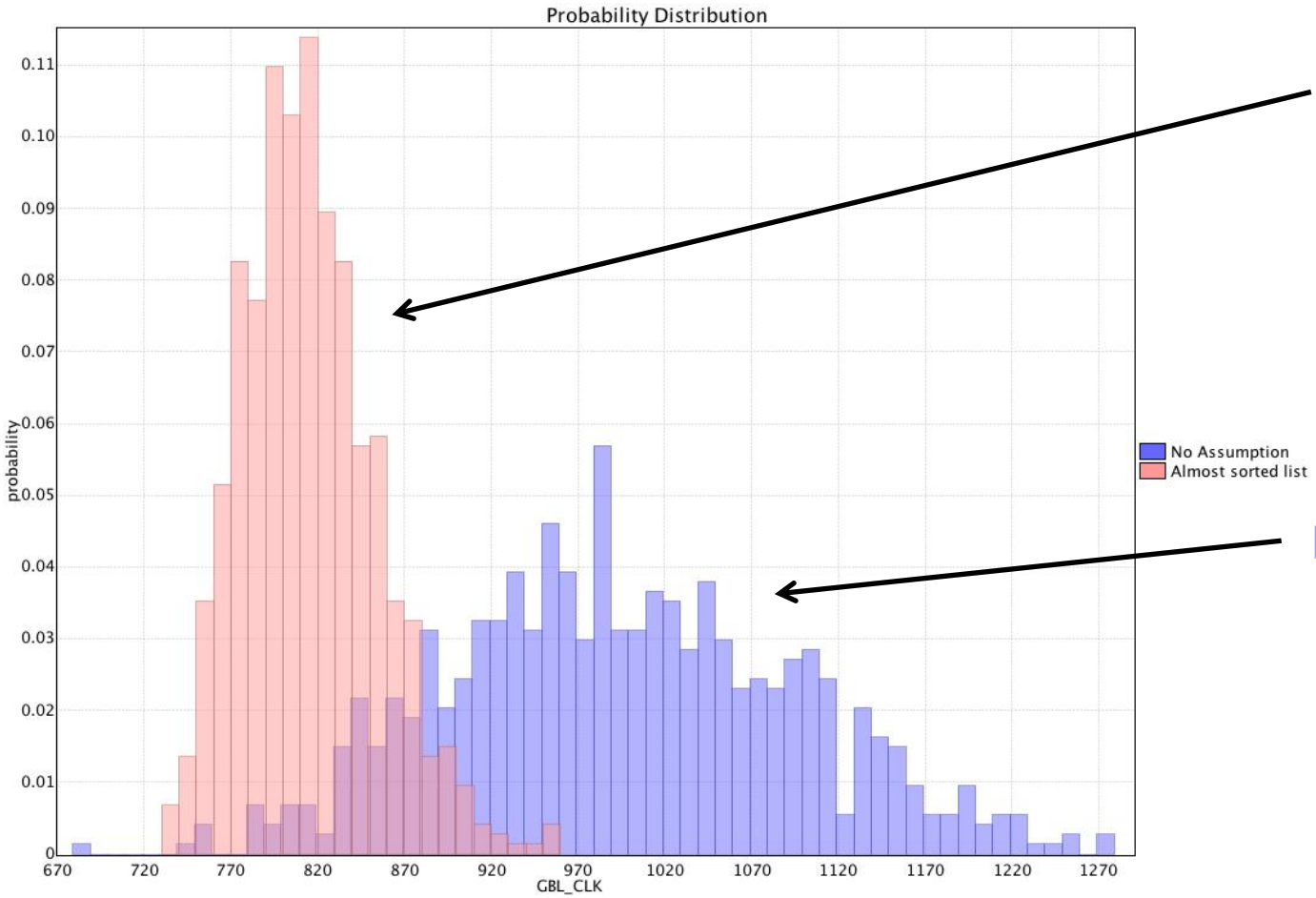
§ Instructions in Slice/Instructions in Program

* Program selected for the WCET Challenge 2006

¶ UPPAAL 4.1.11/Intel Pentium 5/3.1Ghz/16GB

Distribution of Execution Times

UPPAAL Statistical Model Checking: Sorting Algorithm (insertsort-O2)



Almost Sorted List

No Assumption

Conclusion

NICTA

Fully automatic computation of WCET

- Computation of CFG of binary programs + reduced program
  Program slicing
- Formal models of hardware (pipeline and caches)
  Identification of hardware features
- Computation of WCET as a reachability property
  Real-time model-checking with UPPAAL

Experiments to evaluate tightness of results

- method to measure execution-times on ARM920T
- evaluation on benchmarks from Malardalen, Sweden
- over-approximation is less than 5%

Advantages of our method

- Modular
- Fully automatic
- Can easily accommodate new features

Ongoing Work

NICTA

- Support for new architectures
  - Multi-core
  - Assembly languages
- Improved Analysis
  - Interpolant automata for programs
- Computation of most unfavorable initial cache state
  - Interpolant

Evaluation of WCET Techniques and Tools

NICTA

- WCET 2006 challenge

Table 6 Measured and Simulated Execution Times

Nr.	Benchmarks	aiT V1	aiT V2	aiT V3	Chronos V9	Chronos V10	Chronos V11
1	adpcm	buffer	buffer	buffer	160891	183526	126258
2	cnt	19622	16853	7235	4792	5586	3515
3	compress	27308	19970	6824	5859	7504	4744
4	cover	10080	6778	4299	N/A	N/A	N/A
5	crc	buffer	buffer	buffer	22688	26861	18098
6	duff	6919	4610	1028	N/A	N/A	N/A
7	edn	838686	299734	buffer	87444	108973	62995
8	insertsorts	4720	3990	1770	897	1364	949
9	janne_complex	1294	827	359	185	454	356
10	matmult	936602	438435	buffer	186899	185937	90834
11	ndes	401294	190530	buffer	65600	86639	53625
12	ns	73738	36097	buffer	6577	7568	4784
13	nsichneu	28328	18825	8052	6305	42966	40931
14	recursion	8318	7143	5096	N/A	N/A	N/A
15	statemate	2486	3810	1260	1120	6207	5898

N/A = not applicable.
buffer = Because of the buffer limitation, it is not possible to measure the WCETs.

[1] Lili Tan

The Worst Case Execution Time Tool Challenge 2006: The External Test

Leveraging Applications of Formal Methods, pp 241–248, 2006

Evaluation of WCET Techniques and Tools (2)

NICTA

- WCET 2006 challenge

Table 9 Overview of Functional and Service Quality of WCET Tools

Tool	Average Tightness	Benchmarks not Handled by the Tool	Benchmarks Analyzed	Benchmarks under Test	Average Service Rate
aiT	7-8%	0	17	17	100%
Bound-T	N/A	4	13	17	76.5%
SWEET	N/A	2	15	17	88.2%
Chronos	81-89%	4	13	17	76.5%

Tool	Programs Analyzed Without Annotation	Programs Tasks under Test	Average Automation Rate	Complexity of Processor Supported*
aiT	45	84	54%	Simple, Medium, Very Complex
Bound-T	13	51	26%	Simple, Medium
SWEET	15	17	88%	Medium
Chronos	12	51	24%	Configurable Simulated Processor

N/A = No measured WCET was available and no WCET tightness was available at this time.
* = The classification of the processors type is based on the challenge statement.

[1] Lili Tan

The Worst Case Execution Time Tool Challenge 2006: The External Test

Leveraging Applications of Formal Methods, pp 241–248, 2006

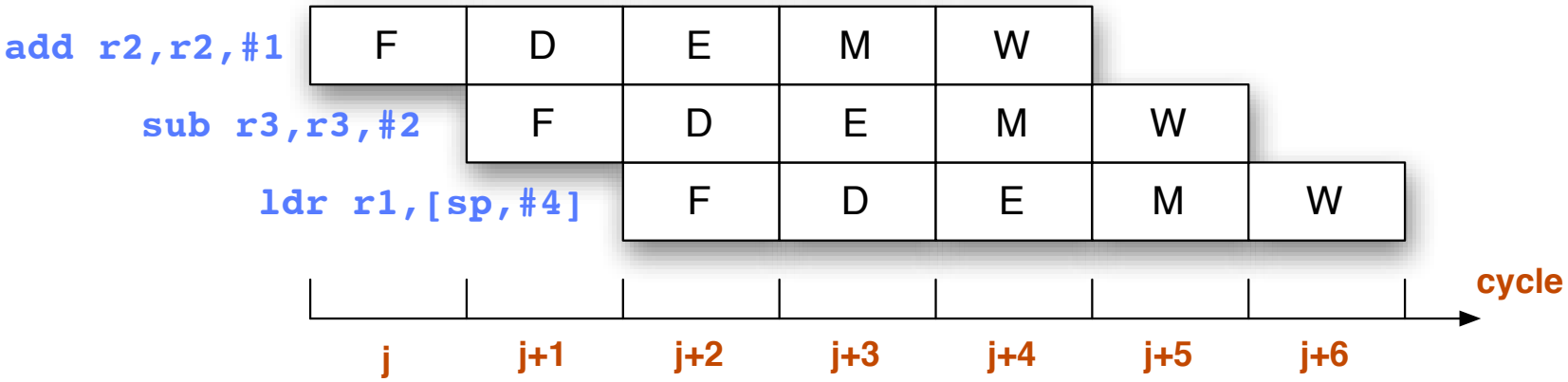
Evaluation of WCET Techniques and Tools (3)

- METAMOC

WCET2010 - LRU

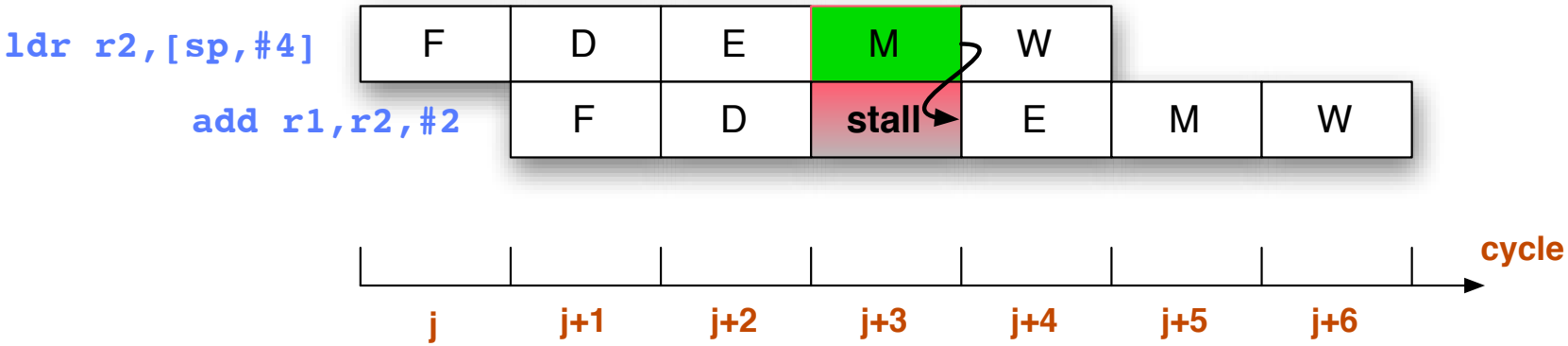
	ATMEL	ARM9	ARM9	ARM9	ARM9	ARM9	ARM9
Optimization	O2	O2	O2	O2	O2	O2	O2
Data-cache	-	miss_writeback	miss_writeback	miss_writeback	Concrete, 64 lines/set, LRU	Concrete, 128 lines/set, LRU	Concrete, LRU
Instr.-cache	-	miss	Concrete, 128 lines/set, LRU	Concrete, LRU	Concrete, LRU	Concrete, LRU	Concrete, LRU
Value-analysis	No	No	No	No	Yes	Yes	Yes
adpcm	**OOM**	**19829472**		**OOM/error**			**OOM/error**
	1:44:19	20:12.95		1:46.67			4:44.83
bs	**146**	**3666**		**964**			**640**
	0:03.66	0:01.09		0:01.06			0:01.27
bsort100		**7997421**		**3425868**	**408715**	**OOM**	**OOM/error**
		0:29.54		0:57.66	2:06.05	2:02.95	1:50.13
cnt	**29572**	**188466**		**52481**			**6137**
	0:02.46	0:02.06		0:02.62			0:04.00
compress	**58352**	**158697**		**69488**		**Model invalid, manual mod.**	**60351**
	6:29.88	0:06.52		0:09.11		0:04.56+0:22.65	0:27.21
crc	**170959**	**2043591**		**328310**			**310722**
	0:36.71	0:13.42		0:26.32			0:39.83
edn	**345499**	**2093937**		**687463**			**431334**
	0:29.42	0:12.81		0:20.56			4:20.71
expint	**7583931**	**652215**		**31483**			**30577**
	4:56.17	0:04.33		0:08.75			0:12.72
fac	**906**	**11553**		**1266**			**683**
	0:01.43	0:01.05		0:01.02			0:01.35
fdct	**4806**	**344234**		**190692**			**168827**
	0:02.43	0:07.10		0:08.26			41:22.70
fibcall	**438**	**13434**		**632**			**599**
	0:01.04	0:01.09		0:01.08			0:01.52
fir	**782494**	**131508**		**21364**			**18763**
	0:25.90	0:02.02		0:02.73			0:04.53
insertsort	**2872**	**75078**		**43888**			**37814**
	0:01.26	0:01.20		0:01.31			0:02.44
janne_complex	**OOM**	**57787**		**1883**			**1883**
	2:11:27	0:01.17		0:01.33			0:01.92
jfdctint	**54383**	**294957**		**133059**			**111510**
	0:05.80	0:05.39		0:06.41			16:46.68
matmult	**525383**	**5836449**		**2636715**		**712869**	**OOM/error**
	0:17.02	0:22.16		0:40.72		1:28.47	1:36.94
ndes	**OOM**	**2575484**	**1084016**	**OOM/error**			**OOM/error**
	26:47.04	2:55.38	7:33.04	8:27.09			4:25.09
ns	**13116**	**279579**		**29060**			**10813**
	0:01.79	0:02.20		0:03.43			0:07.09
nsichneu		**1020957**		**622904**			**295997**
		3:04.11		3:25.56			3:37.70
prime	**170806**	**6477669**		**470372**			**469060**
	0:12.52	1:02.36		1:56.36			2:41.87
ud	**58217**	**1160049**		**175359**			**165814**
	0:11.81	0:07.76		0:14.64			0:23.07
	3 errors / 19 benchmarks	0 errors / 21 benchmar	1 errors / 21 benchmarks	errors / 21 benchmarks	2 errors / 21 benchmarks	3 errors / 21 benchmarks	4 errors / 21 benchmarks
	Model checking fails: 3		Model checking fails: 1 Manual modification: 1	Model checking fails: 2 Manual modification: 0	Value analysis fails: 0 Model checking fails: 2 Manual modification: 3	Value analysis fails: 0 Model checking fails: 3 Manual modification: 2	Value analysis fails: 0 Model checking fails: 4 Manual modification: 1

Pipelining



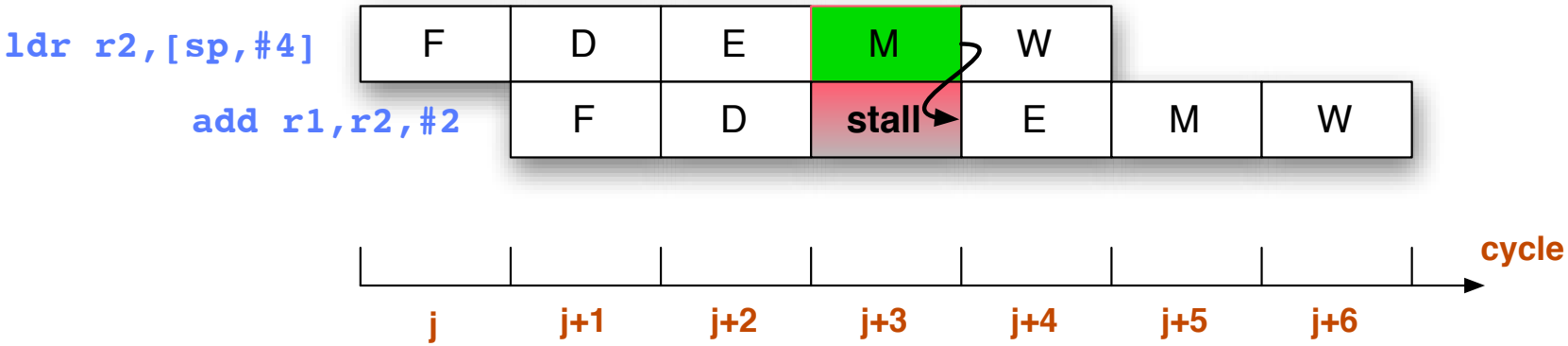
Pipeline Stalls

- Data dependences

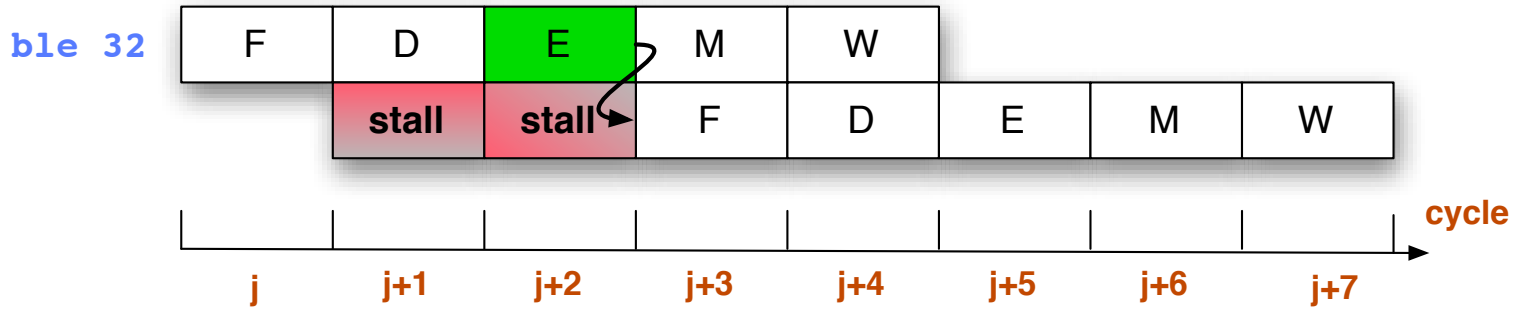


Pipeline Stalls

- Data dependences



- Dynamic computation of address of next instruction



Caches