Real-Time and Embedded Systems

Lab 6

Alex Hoffman
alex.hoffman@tum.de

TU München
Institute for Real-Time Computer Systems (RCS)

2020

Lehrstuhl für Realzeit-Computersysteme

WCET Analysis – Overview

- **Goal:**
  - Determine how long your code takes in the worst case to complete a certain procedure (e.g., one C function)

WCET Analysis – Overview

- **Goal:**
  - Determine how long your code takes in the worst case to complete a certain procedure (e.g., one C function)
- **Why?**
  - verify the *synchrony hypothesis*
  - to perform a *schedulability analysis* (when multiple programs run in parallel: does *each* of them finish in time?)

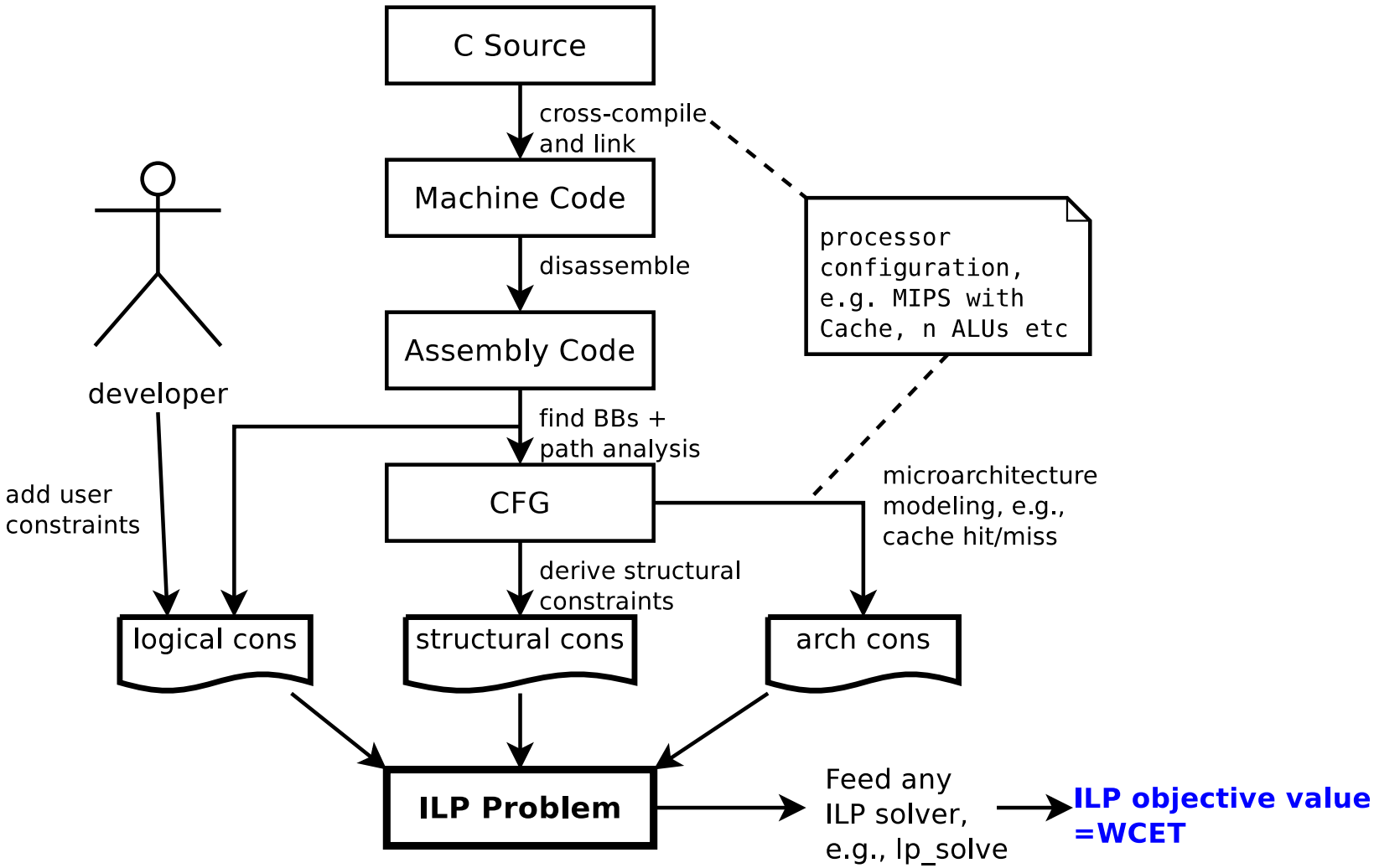
WCET Analysis – Overview

- **Goal:**
  - Determine how long your code takes in the worst case to complete a certain procedure (e.g., one C function)
- **Why?**
  - verify the *synchrony hypothesis*
  - to perform a *schedulability analysis* (when multiple programs run in parallel: does *each* of them finish in time?)
- WCET estimation = hard problem to solve
- there is only a "handful" of people who provide tools
  - NUS' *Chronos* (academic)
  - AbsInt's *WCET Analyzer* (used by Airbus)
  - Tidorum's *Bound-T* (used by ESA)
  - some lesser known...

WCET Analysis – Overview

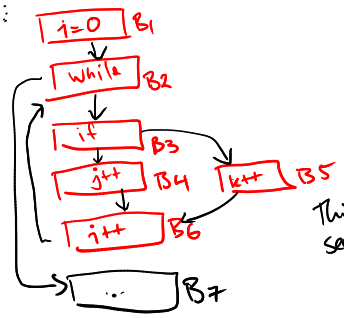
1. **cross-compile to assembler code**
2. **find basic blocks (BB)**: group instructions that are always executed together
3. **generate control flow graph (CFG)**: directed graph derived from assembly.
4. **derive structural constraints**: read-off from CFG
5. **add user constraints:** e.g., logical constraints. "This loop will not run more than x times."
6. **determine execution time of each BB**
7. **Feed ILP solver**:
   - objective fcn $(\sum_i c_i x_i)$ + constraints go in
   - the block counts (x_i) and resulting WCET come out

WCET Analysis – Overview



What possible values could 'ok' have before X_1?

```
x1   i = 0;
x2   while (i < 100) {
x3     if (ok) {
x4       j++;
     } else {
x5       k++; ok = true;
x6     i++;
x7   }
```

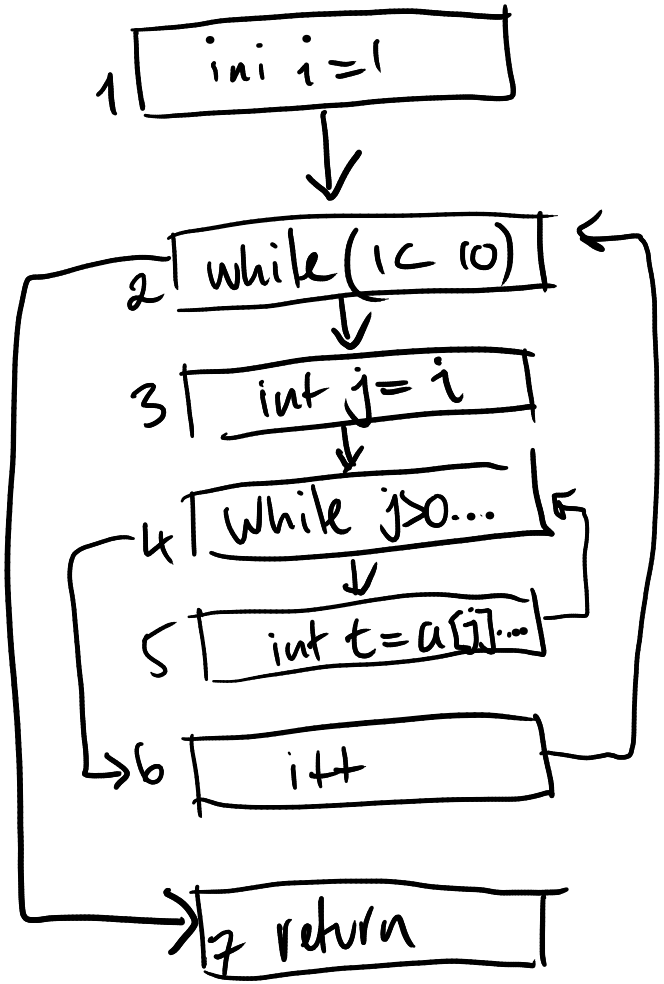


True	False
$X_4 = 100$	$X_4 = 99$
$X_5 = 0$	$X_5 = 1$
∴ We know	$X_5 <= 1$
	$100 >= X_4 >= 99$

This seq → B_1, B_2, B_7 → would give $X_4 = 0$

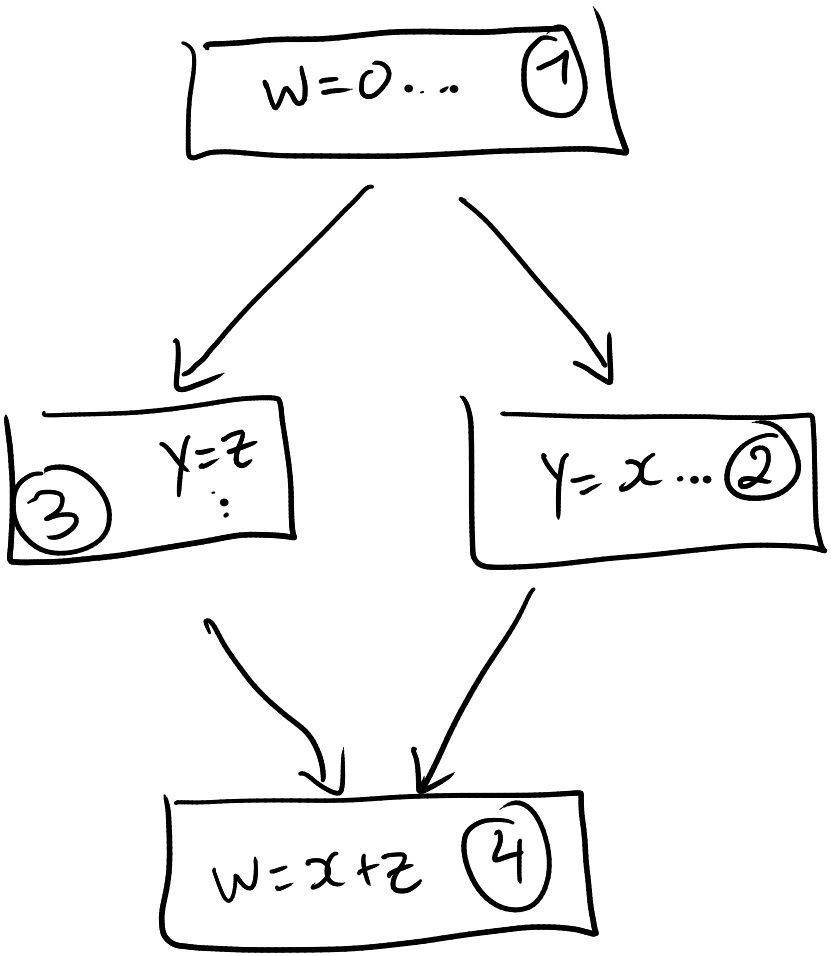
This is not allowed/ valid

WCET Analysis – Example: Sorting




```
int sort(int a[10]) {
  int i = 1;
  while(i < 10)
  {
    int j = i;
    while( j>0 &&
           a[j-1] > a[j])
    {
      int t = a[j];
      a[j] = a[j-1];
      a[j-1] = t;
      j--;
    }
    i++;
  }
  return i;
}
```

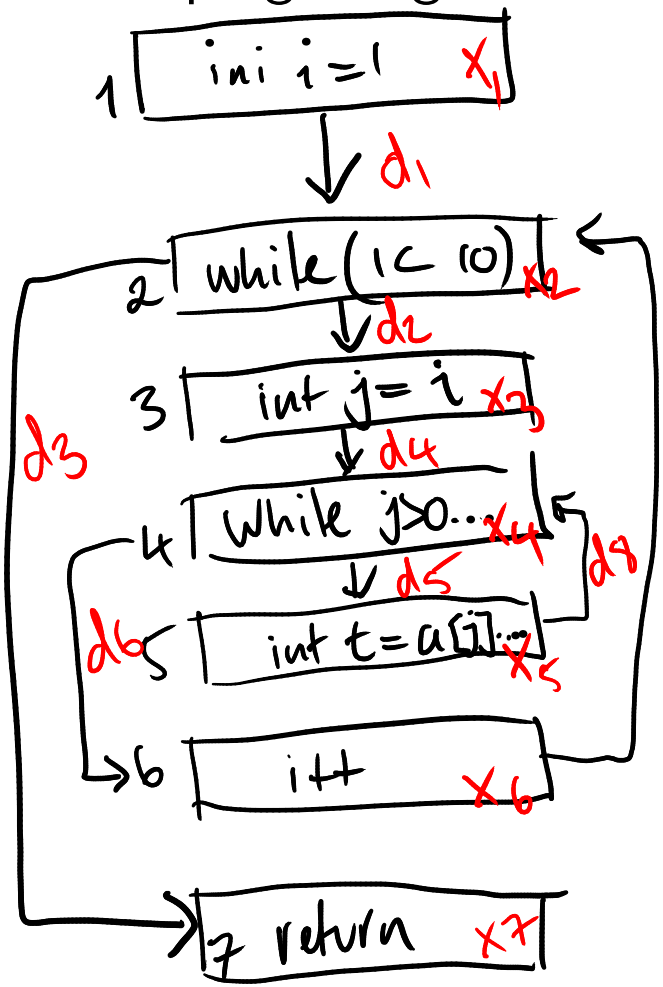
WCET Analysis – Example: Sorting




```
w = 0;          1
x = x + y;
if( x > z ){
    y = x;      2
    x++;
} else {
    y = z;      3
    z++;
}
w = x + z;      4
```

WCET Analysis – Structural Constraints

Draw the CFG and write down the structural constraints for the sort program given before. x_1 comes from hardware model



① $x_1 = d_1$

② $x_2 = d_1 + d_7 = d_2 + d_3$

③ x_1 is some set of instructions ∴ it has some — exec time

④ $x_3 = d_2$

⑤ $x_3 = d_4$

⑥ $x_4 = d_4 + d_8 = d_6 + d_5$

⑦

⑧ $x_5 = d_5 = d_8$

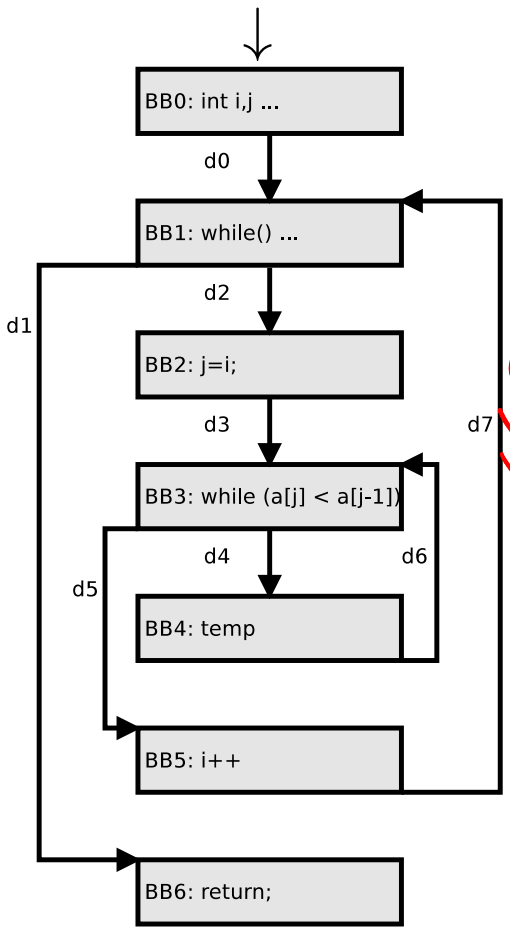
$x_6 = d_6 = d_7$

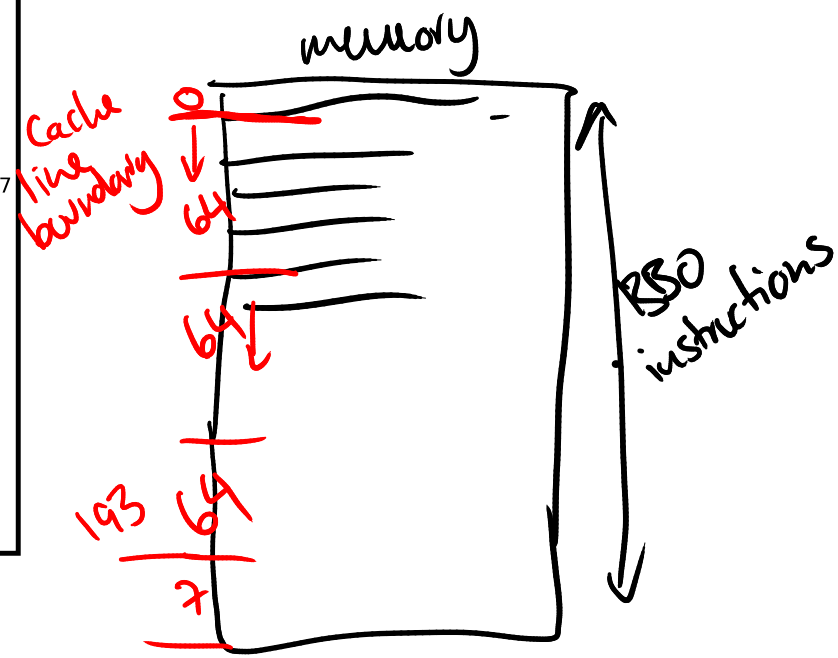
$x_7 = d_3$

WCET Analysis – Structural Constraints

Draw the CFG and write down the structural constraints for the sort program given before.

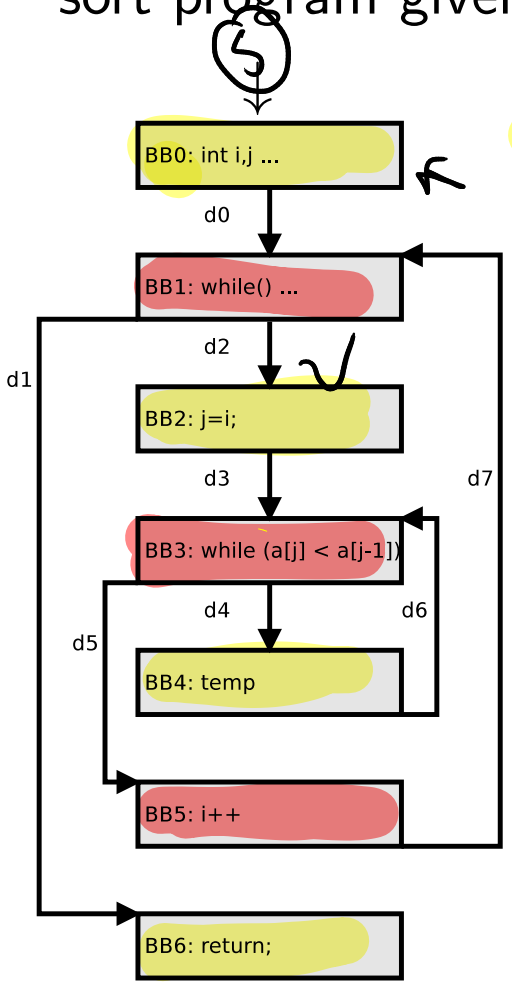
Eg. If BB0 had 200 Bytes of instructions in it and your system had 64 byte cache lines then you could need up to 5 cache lines to load BB0

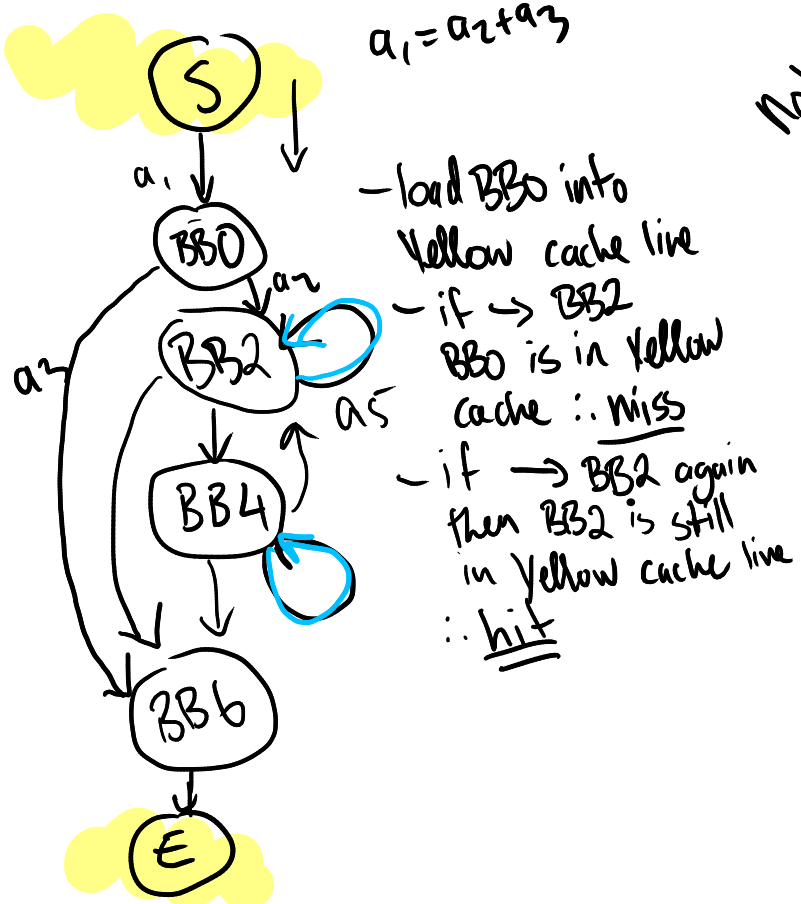


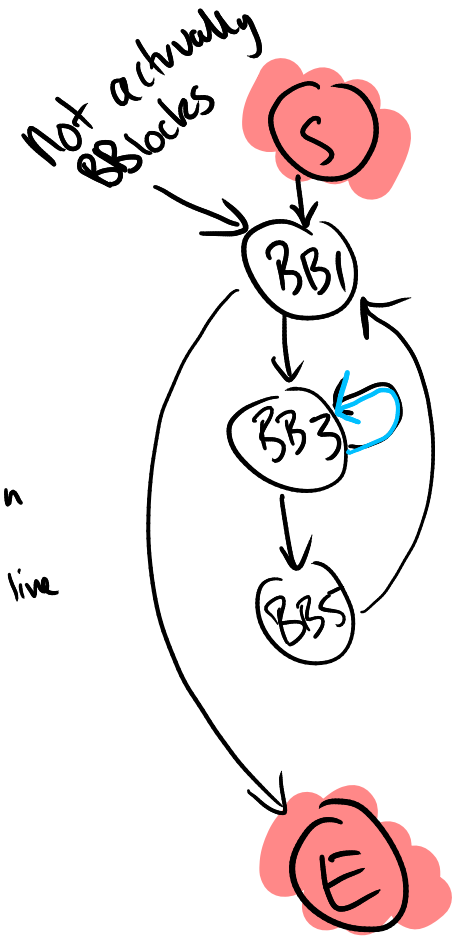


WCET Analysis – Structural Constraints

Draw the CFG and write down the structural constraints for the sort program given before. For this example BBlock == L block

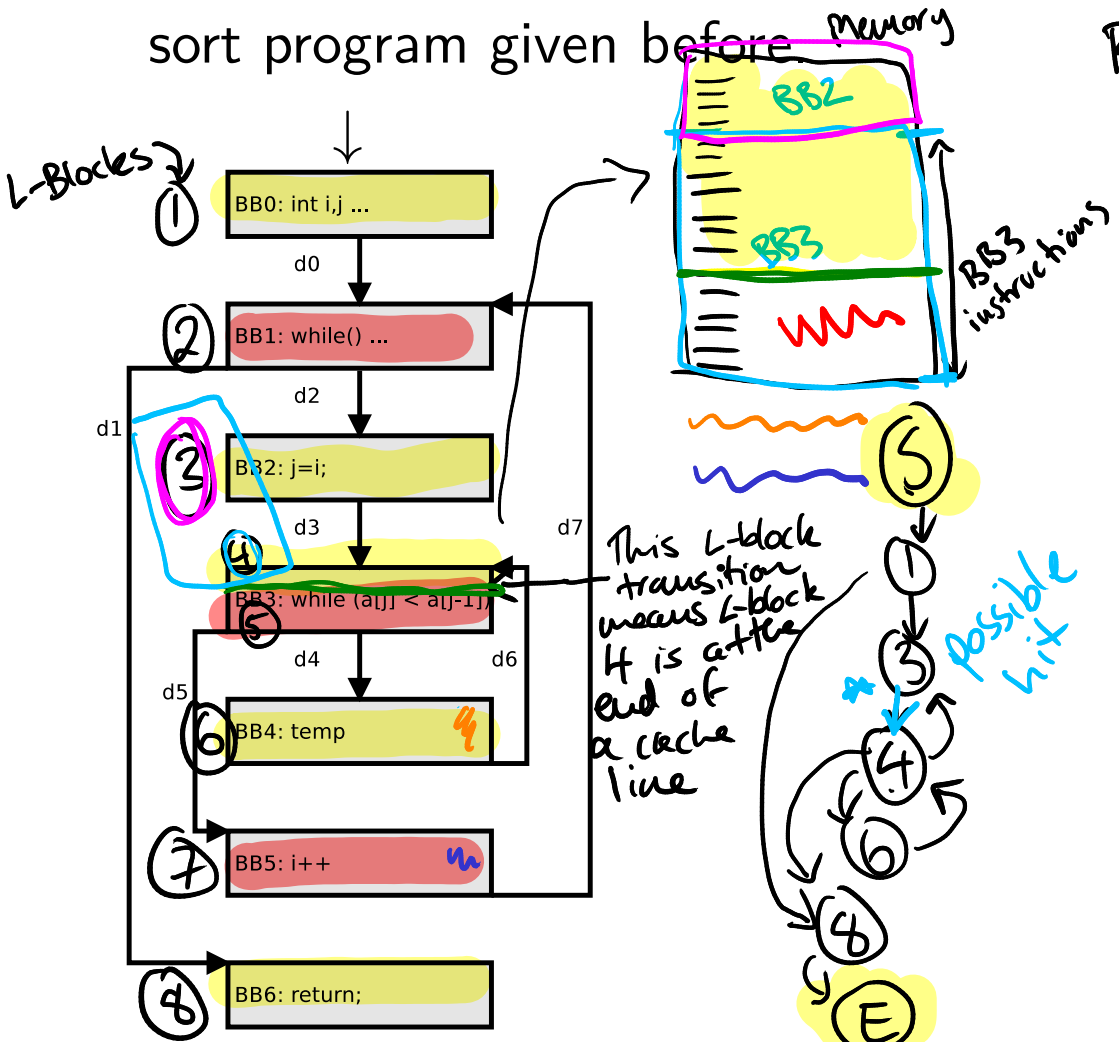






WCET Analysis – Structural Constraints

Draw the CFG and write down the structural constraints for the sort program given before.

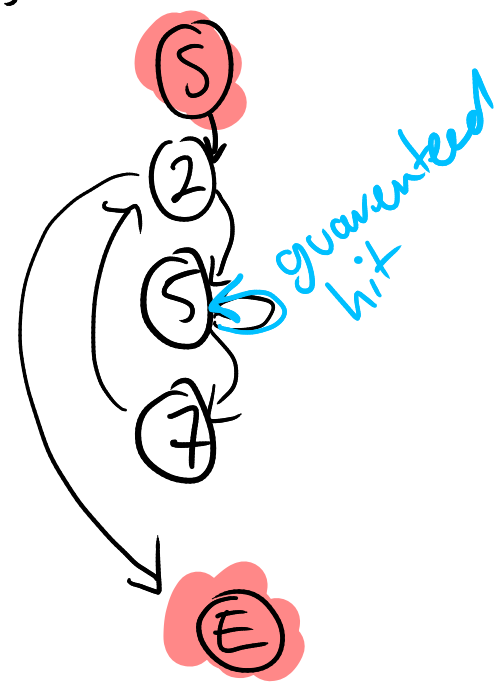


BB3 uses 2 cache lines

∴ BB3 is comprised of 2 L-blocks

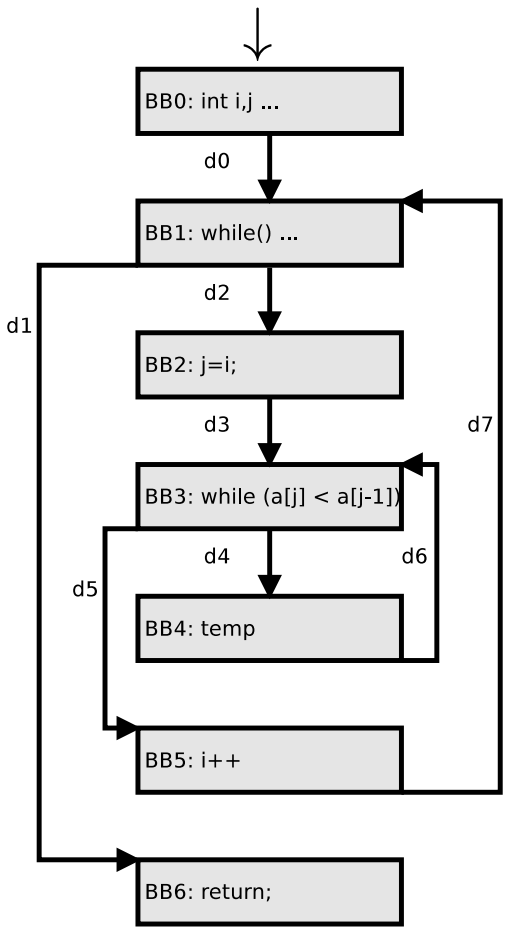
1 is for yellow cache

1 is for red cache



WCET Analysis – Structural Constraints

Draw the CFG and write down the structural constraints for the sort program given before.



$$x_0 = d_0 = 1 \quad (1)$$

$$x_1 = d_0 + d_7 = d_1 + d_2 \quad (2)$$

$$x_2 = d_2 = d_3 \quad (3)$$

$$x_3 = d_3 + d_6 = d_4 + d_5 \quad (4)$$

$$x_4 = d_4 = d_6 \quad (5)$$

$$x_5 = d_5 = d_7 \quad (6)$$

$$x_6 = d_1 \quad (7)$$

WCET Analysis – Logical Constraints

- This program contains a loop
- We have to help the analyzer, by telling it how often the loop may iterate
- so we have to find the *loop bounds* and give them to the analyzer
- **outer:** $= 9$
- **inner:** $\leq j - 1$
- **total:** ≤ 45 : $\left(1 + 2 + \cdots + 9 = \frac{9+10}{2}\right)$

```
int sort(int a[10]) {
  int i = 1;
  while(i < 10)
  {
    int j = i;
    while( j>0 &&
           a[j-1] > a[j])
    {
      int t = a[j];
      a[j] = a[j-1];
      a[j-1] = t;
      j--;
    }
    i++;
  }
  return i;
}
```