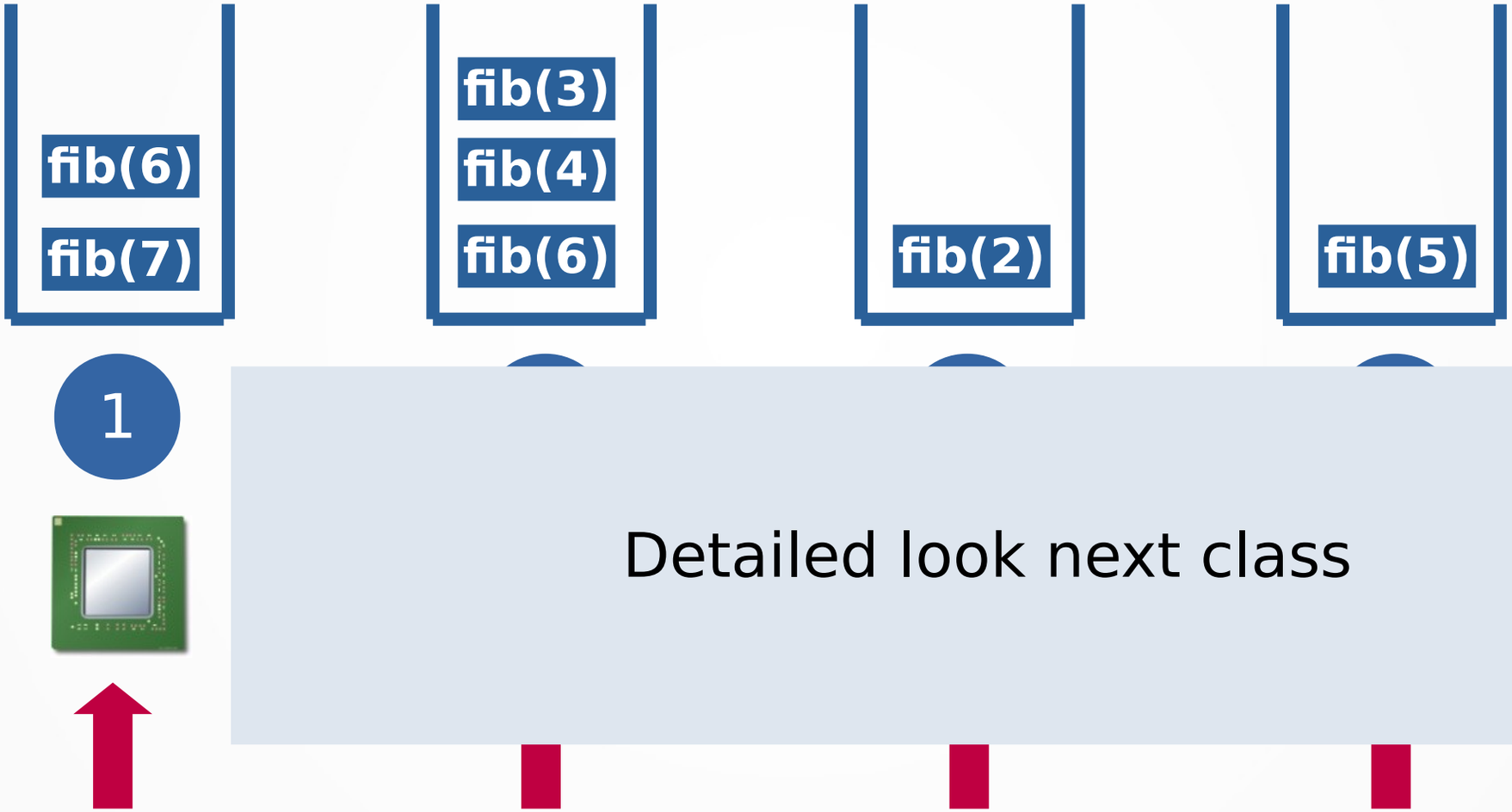
Cilk Compiler and Runtime

Dynamic Load Balancing



Detailed look next class

Mastery of (CS) material

- 1. Know how to **use** a framework, language, library
- 2. Understand **design choices** / **goals** of a framework
- 3. Understand **implementation details**
- 4. Decipher literal **code** implementation
- 5. **Implement** it yourself

Mastery of (CS) material

- 1. Know how to **use** a framework, language, library
- 2. Understand **design choices / goals** of a framework
- 3. Understand **implementation details**
- 4. Decipher literal **code** implementation
- 5. **Implement** it yourself

Mastery of (CS) material

- 1. Know how to **use** a framework, language, library
- 2. Understand **design choices / goals** of a framework
- 3. Understand **implementation details**
- 4. Decipher literal **code** implementation
- 5. **Implement** it yourself

Mastery of (CS) material

- 1. Know how to **use** a framework, language, library
- 2. Understand **design choices / goals** of a framework
- 3. Understand **implementation details**
- 4. Decipher literal **code** implementation
- 5. **Implement** it yourself

Mastery of (CS) material

- 1. Know how to **use** a framework, language, library
- 2. Understand **design choices / goals** of a framework
- 3. Understand **implementation details**
- 4. Decipher literal **code** implementation
- 5. **Implement** it yourself

Mastery of (CS) material

- 1. Know how to **use** a framework, language, library
- 2. Understand **design choices / goals** of a framework
- 3. Understand **implementation details**
- 4. Decipher literal **code** implementation
- 5. **Implement** it yourself

Great start!

Mastery of (CS) material

- 1. Know how to **use** a framework, language, library
- 2. Understand **design choices / goals** of a framework
- 3. Understand **implementation details**
- 4. Decipher literal **code** implementation
- 5. **Implement** it yourself

Important for serious development

Mastery of (CS) material

- 1. Know how to **use** a framework, language, library
- 2. Understand **design choices / goals** of a framework
- 3. Understand **implementation details**
- 4. Decipher literal **code** implementation
- 5. **Implement** it yourself

Separates you from the next person

Mastery of (CS) material

- 1. Know how to **use** a framework, language, library
- 2. Understand **design choices / goals** of a framework
- 3. Understand **implementation details**
- 4. Decipher literal **code** implementation
- 5. **Implement** it yourself

Really separates you
From the next person

Mastery of (CS) material

- 1. Know how to **use** a framework, language, library
- 2. Understand **design choices / goals** of a framework
- 3. Understand **implementation details**
- 4. Decipher literal **code** implementation
- 5. **Implement** it yourself

Really separates you
From the next person

Real-world projects grow incredibly complex;
Inspecting them will teach you so much!

Real-world projects grow incredibly complex;
Inspecting them will teach you so much!

Everytime I look at a famous project for the first
Time (gcc, golang, clang) I feel like a beginner.

Work Stealing

- Fib example on board

Compiling Cilk Function

```
int f(int n) {
  Int x, y;
  x = cilk_spawn g(n);
  Y = h(n);
  cilk_sync;
  return x + y;
}
```

```
int f(int n) {
  __cilkrts_stack_frame f_sf;
  __cilkrts_worker* w = __cilkrts_get_tls_worker();
  f_sf.call_parent = w->current_stack_frame;
  f_sf.worker = w;
  w->current_stack_frame = &f_sf;

  int x, y;
  if (!CILK_SETJMP(f_sf.ctx))
    _cilk_spawn_helper_g(&x, n);

  y = h(n);

  if (f_sf.flags & CILK_FRAME_UNSYNCHED)
    if (!CILK_SETJMP(f_sf.ctx))
      __cilkrts_sync(f_sf);
  __cilkrts_pop_frame(&f_sf);
  if (f_sf.flags)
    __cilkrts_leave_frame(f_sf);
}
```

Compiling Cilk Function

```
int f(int n) {
  Int x, y;
  x = cilk_spawn g(n);
  Y = h(n);
  cilk_sync;
  return x + y;
}
```

```
int f(int n) {
  __cilkrts_stack_frame f_sf;
  __cilkrts_worker* w = __cilkrts_get_tls_worker();
  f_sf.call_parent = w->current_stack_frame;
  f_sf.worker = w;
  w->current_stack_frame = &f_sf;

  int x, y;
  if (!CILK_SETJMP(f_sf.ctx))
    _cilk_spawn_helper_g(&x, n);

  y = h(n);

  if (f_sf.flags & CILK_FRAME_UNSYNCHED)
    if (!CILK_SETJMP(f_sf.ctx))
      __cilkrts_sync(f_sf);
  __cilkrts_pop_frame(&f_sf);
  if (f_sf.flags)
    __cilkrts_leave_frame(f_sf);
}
```

Fast clone skip sync

Compiling Cilk Function

```
int f(int n) {
  Int x, y;
  x = cilk_spawn g(n);
  Y = h(n);
  cilk_sync;
  return x + y;
}
```

```
int f(int n) {
  // ...
  if (!CILK_SETJMP(f_sf.ctx))
    _cilk_spawn_helper_g(&x, n);
  // ...
}

void cilk_spawn_helper_g(int* x, int n) {
  __cilkrts_stack_frame g_hf;
  __cilkrts_enter_frame_fast(&g_hf);

  // Evaluate arguments.
  // Nothing to do in this example.

  __cilkrts_detach();

  *x = g(n);

  __cilkrts_pop_frame(&g_hf);
  if (g_hf.flags)
  __cilkrts_leave_frame(g_hf);
}
```

Compiling Cilk Function

```
int f(int n) {
  Int x, y;
  x = cilk_spawn g(n);
  Y = h(n);
  cilk_sync;
  return x + y;
}
```

```
int f(int n) {
  // ...
  if (!CILK_SETJMP(f_sf.ctx))
    _cilk_spawn_helper_g(&x, n);
  // ...
}

void cilk_spawn_helper_g(int* x, int n) {
  __cilkrts_stack_frame g_hf;
  __cilkrts_enter_frame_fast(&g_hf);

  // Evaluate arguments.
  // Nothing to do in this example.

  __cilkrts_detach();

  *x = g(n);

  __cilkrts_pop_frame(&g_hf);
  if (g_hf.flags)
  __cilkrts_leave_frame(g_hf);
}
```

Work-First Policy

Compiling Cilk Function

```
int f(int n) {
  Int x, y;
  x = cilk_spawn g(n);
  Y = h(n);
  cilk_sync;
  return x + y;
}
```

```
int f(int n) {
  // ...
  if (!CILK_SETJMP(f_sf.ctx))
    _cilk_spawn_helper_g(&x, n);
  // ...
}

void cilk_spawn_helper_g(int* x, int n) {
  __cilkrts_stack_frame g_hf;
  __cilkrts_enter_frame_fast(&g_hf);

  // Evaluate arguments.
  // Nothing to do in this example.

  __cilkrts_detach();

  *x = g(n);

  __cilkrts_pop_frame(&g_hf);
  if (g_hf.flags)
  __cilkrts_leave_frame(g_hf);
}
```

On exit, check for parent steal

Implementation

- Gcc 7.4 release (https://github.com/gcc-mirror/gcc)
  - Code contained in gcc/libcilkrts
- Files to look at:
  - runtime/cilk_abi.c
  - runtime/scheduler.c
- Import Data
  - __cilkrts_stack_frame
  - full_frame
  - __cilkrts_worker
- Important Func
  - enter_frame_internal
  - __cilkrts_undo_detach
  - __cilkrts_leave_frame
  - __cilkrts_sync

Acknowledgements

- Yu David Liu (Cilk Lecture Notes)
- Jim Sukha (Cilk Runtime Notes)