Intermediate Representation Design Considerations

Sorav Bansal

Abstraction Example

$$x = a + b$$

$$y = x - a$$

$$x[0] = a[0] \oplus b[0]$$

$$x[1] = a[0]\, b[0] \oplus a[1] \oplus b[1]$$

$$x[2] = \ldots$$

$$y[0] = a[0] \oplus b[0]$$

$$y[1] = \ldots$$

Abstraction Example

```
for (i = 0 ... n)
    body

print ...;  // data
            // independent
```

```
i = 0;
while (i <= n)
    body
    i++
print ...;
```

want to hoist print ...

Abstraction Example

```
list<int> l1, l2;
...
sz = l1.size();
l2.push_back(0);
return l1.size()==sz;
```

```
struct node {
  int e; node *next;
}*l1=null, *l2=null;
...
for(sz=0, cur=l1;
    *cur;
    cur=cur->next,sz++);
for(cur=l2; cur->next;
    cur=cur->next);
cur->next=new...
...
```

Abstraction Example

```
Tensor<float> t1, t2;
...
t3 = t1 * t2;
t4 = inverse(t1);
return t4 * t3;
```

```
float t1[..][..]
      t2[..][..];
...
for(...)
  for(...)
    for(...)
      ...
      ...
      ...
```

IR Example

x = y + z
s = a + b
t = x - 3
y = f(y)
...
(3 Address Code)

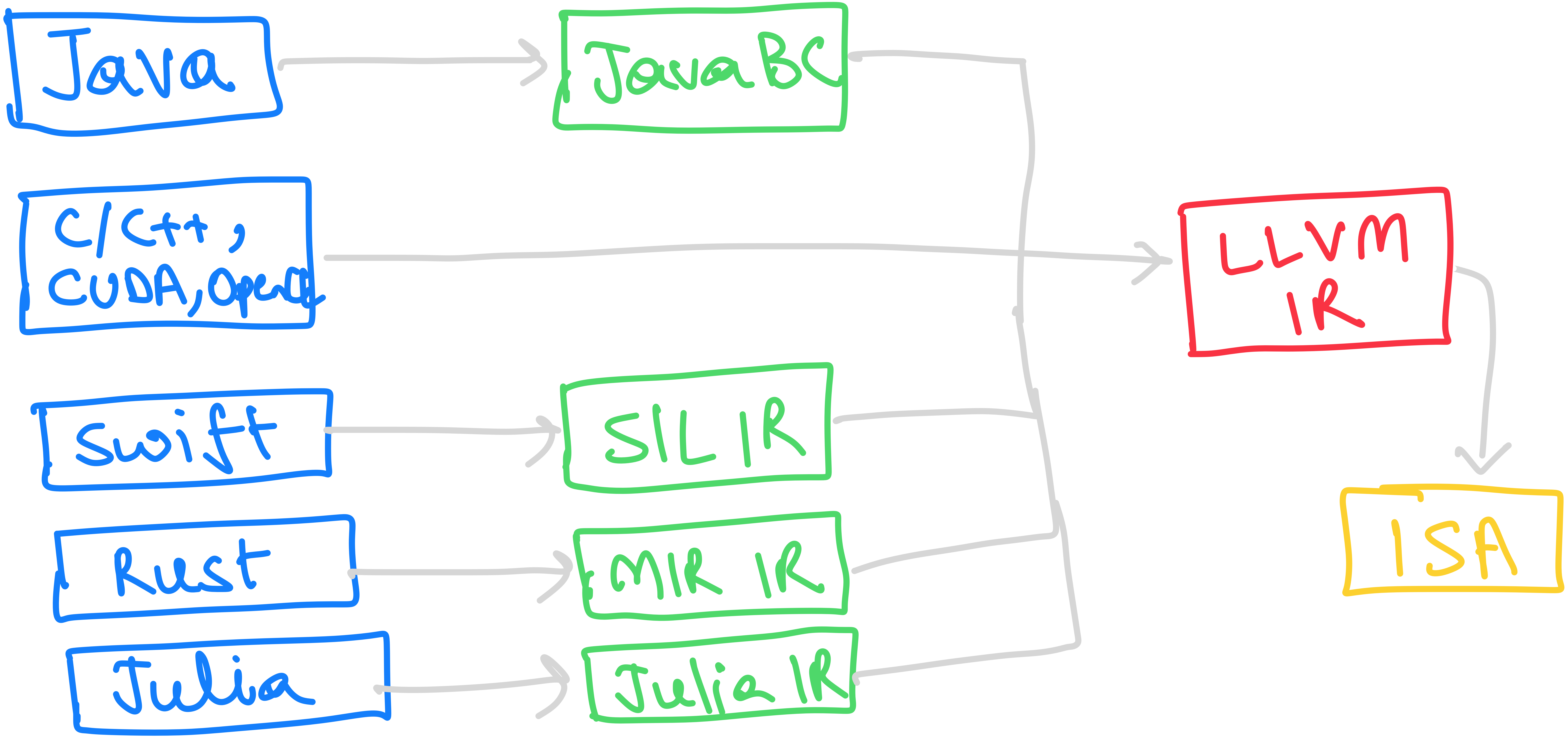
r1 = [sp+4] + [bp-4]
[sp+8] = [bp-8] + r2
[sp+12] = r1 - 3
push [sp+4]
call f
pop
[sp+4] = r0 ...

IR Example

$$x = \langle 4 \times i8 \rangle\, y + \langle 4 \times i8 \rangle\, z$$

$$x0 = y0 + z0$$
$$x1 = y1 + z1$$
$$x2 = y2 + z2$$
$$x3 = y3 + z3$$

Compilation Pipeline Examples



Machine Learning and Why Its Compilation is an Important Problem

1. Neural Networks have proven promise for some applications

implement function fitting in a reasonably general way

Machine Learning and Why Its Compilation is an Important Problem

1. Neural Networks have proven promise for some applications
2. Compute Hungry

Lots of number crunching

Machine Learning and Why Its Compilation is an Important Problem

1. Neural Networks have proven promise for some applications
2. Compute Hungry
3. Traditional CPU abstractions too inefficient

Data Parallel

Regular

Example Hardware

Tensor cores in nVidia GPUs

Parallel mixed-precision matrix multiply and accumulate instructions

Example Hardware

MatMul unit (MXU)

Large on-chip cache

Intel NNP

Qualcomm Cloud AI 100

Google TPU

Amazon Inferentia

Example Programming Models

TensorFlow
PyTorch
...
} through common PL interfaces

Dataflow graph of primitive operators

→ similar to a control flow graph (CG) except with explicit parallelism

NN Compilers

High-Level IR

DAG based IR

- Primitive ops mostly
- Cycles are typically unrolled
- Data typically represented as n-dim arrays (tensors)

NN Compilers

High-Level IR

DAG based IR

Rewrite rule based transformations

(peephole optimization)

DFA based transformations

NN Compilers

Lower-level IR

e.g., MLIR dialects

Polyhedral-based IR

Polyhedral analysis will be the discussion subject during the first part of the course

MLIR Example

```
// Affine loops are Ops with regions.
affine.for %arg0 = 0 to %N {
  // Only loop-invariant values, loop iterators, and affine functions of
  // those are allowed.
  affine.for %arg1 = 0 to %N {
    // Body of affine for loops obey SSA.
    %0 = affine.load %A[%arg0] : memref<? x f32>
    // Structured memory reference (memref) type can have
    // affine layout maps.
    %1 = affine.load %B[%arg1] : memref<? x f32, (d0)[s0] -> (d0 + s0)>
    %2 = mulf %0, %1 : f32
    // Affine load/store can have affine expressions as subscripts.
    %3 = affine.load %C[%arg0 + %arg1] : memref<? x f32>
    %4 = addf %3, %2 : f32
    affine.store %4, %C[%arg0 + %arg1] : memref<? x f32>
  }
}
```

Viewing for this week

- Compiler Design Modules 125 - 130
- https://iitd.github.io/col874