© Manuel Cargaleiro

Overview of Intermediate Representations (IRs)

Compilers course

Masters in Informatics and Computing Engineering (MIEIC), 3rd Year

U.PORTO

João M. P. Cardoso

Universidade do Porto
FEUP Faculdade de Engenharia

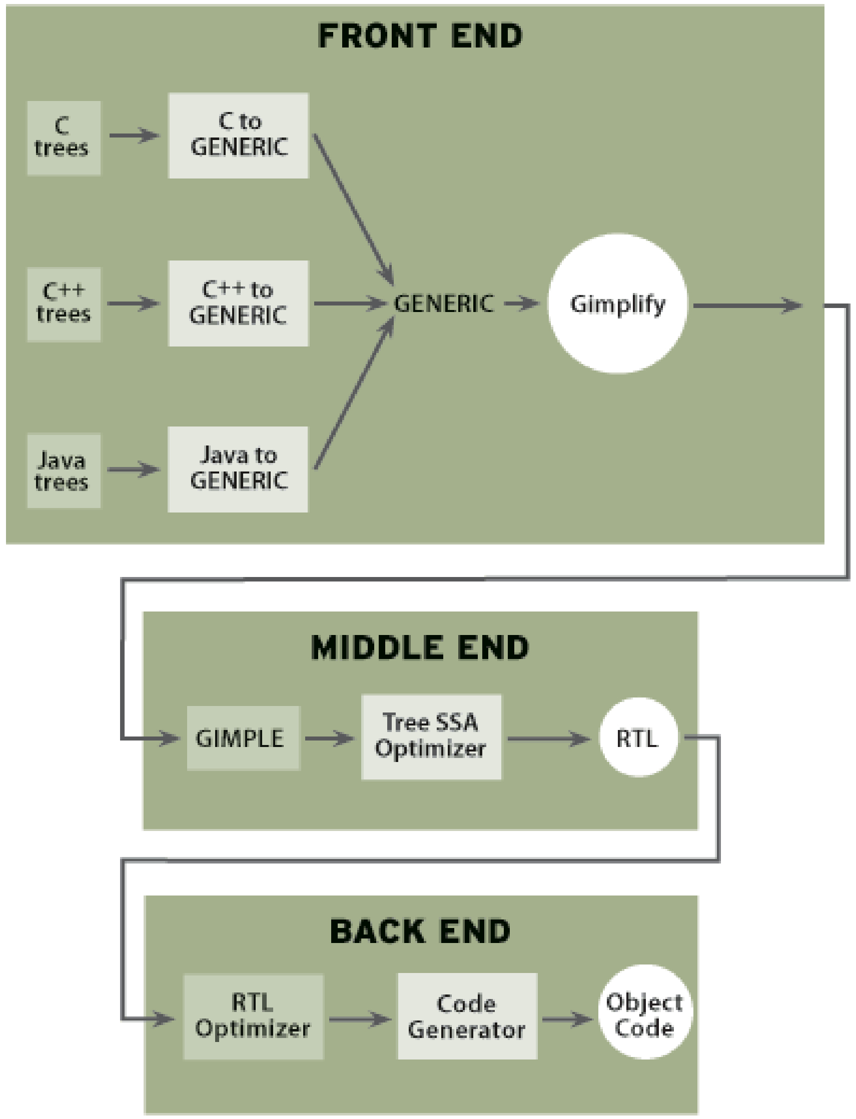
Dep. de Engenharia Informática
Faculdade de Engenharia (FEUP), Universidade do Porto,
Porto, Portugal
Email:jmpc@acm.org

Intermediate Representations

GCC EXAMPLE

GNU GCC

- **GNU Compiler Collection** (usually shortened to **GCC**)
  - GENERIC (language independent tree structure used by the front-ends)
  - GIMPLE (High-Level, Low-Level and Low-Level in SSA form)
    - http://gcc.gnu.org/wiki/GIMPLE
  - RTL (**register transfer language**), which looks like a Lisp S-expression:
    - (set:SI (reg:SI 140) (plus:SI (reg:SI 138) (reg:SI 139)))
    - Equivalent to: Reg 140 ← Reg 138 + Reg 139;



https://gcc.gnu.org/onlinedocs/gccint/

GNU GCC: Gimple

- Arithmetic Expressions:
  - a = b + c + d
- Becomes:

```
T1 = b + c;
a = T1 + d;
```

- Conditional Expressions
  - a = b ? c : d;
- Becomes:

```
if (b)
    T1 = c;
else
    T1 = d;
a = T1;
```

GNU GCC: RTL

- Three address code
- Example:
  - (set:SI (reg:SI 140) (plus:SI (reg:SI 138) (reg:SI 139)))
  - Equivalent to: Reg 140 ← Reg 138 + Reg 139;
- (reg:m n),
  - *n*: is a register (hard or pseudo)
  - *m* : is the machine mode of the reference

https://gcc.gnu.org/onlinedocs/gccint/RTL.html#RTL

Low-Level Intermediate

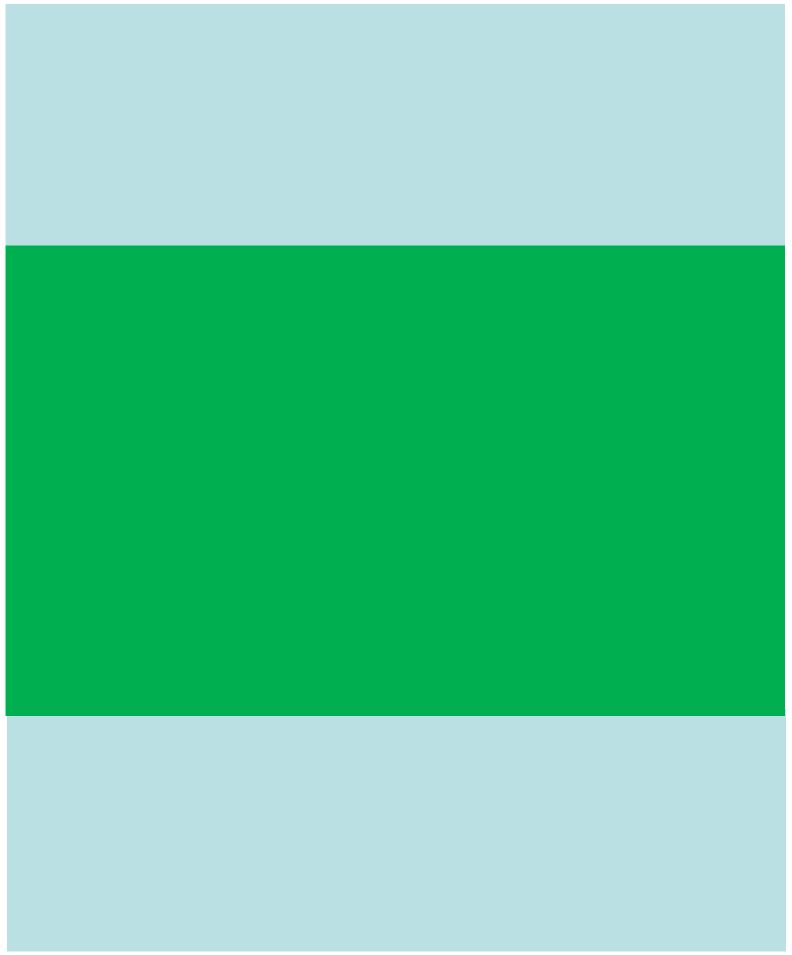
THREE-ADDRESS CODE

Example

```
int dotprod(int x[], int y[], int nx)
{
    int sum = 0, i;

    for (i = 0; i < nx; i++)
        sum += x[i] * y[i];

    return sum;
}
```



Example

```
int DSP_dotprod_c(const short *x,
     const short *y, int nx)
{
    int sum = 0, i;

    for (i = 0; i < nx; i++)
        sum += x[i] * y[i];

    return sum;
}
```

```
        sum=0;
        i=0;
cont_l: If(i >= nx) goto end_l;
        t1=load x, i;
        t2=load y, i;
        t3=t1+t2;
        sum = sum +t3;
        i=i+1;
        goto cont_l;
end_l:
        return sum;
```

Example

```
int DSP_dotprod_c(const short *x,
     const short *y, int nx)
{
   int sum = 0, i;

   for (i = 0; i < nx; i++)
       sum += x[i] * y[i];

   return sum;
}
```

```
           sum=0;
           i=0;
cont_l:    If(i >= nx) goto end_l;
           a1=4*i+x;
           t1=load a1;
           a2=4*i+y;
           t2=load a2;
           t3=t1+t2;
           sum = sum +t3;
           i=i+1;
           goto cont_l;
end_l:
           return sum;
```

Example

```
        sum=0;
        i=0;
cont_l: If(i >= nx) goto end_l;
        a1=4*i+x;
        t1=load a1;
        a2=4*i+y;
        t2=load a2;
        t3=t1+t2;
        sum = sum +t3;
        i=i+1;
        goto cont_l;
end_l:
        return sum;
```

```
        iconst 0
        istore 3 // sum =0;
        iconst 0
        istore 4  // i=0;
cont_l: If(i >= nx) goto end_l;
        a1=4*i+x;
        t1=load a1;
        a2=4*i+y;
        t2=load a2;
        t3=t1+t2;
        sum = sum +t3;
        i=i+1;
        goto cont_l;
end_l:
        return sum;
```