

# Retargeting LLVM to an Explicit Data Graph Execution (EDGE) Architecture

EuroLLVM Barcelona

March 18, 2016

Aaron Smith

# Explicit Data Graph Execution (EDGE)

- ISA and microarchitecture invented at UT-Austin in early 2000's in the TRIPS project
  - Hybrid von Neumann/Dataflow execution model
    - Iannucci and Arvind @ MIT in late 80's
  - Block Structured ISAs
    - Yale Patt et al. @ Michigan in mid 90's
- Several Advantages
  - Out-of-order execution with near in-order power efficiency
  - Core fusion to accelerate single threads
  - Supports imperative languages such as C/C++ unlike previous dataflow machines
    - porting is a recompile not a rewrite
- But no open source toolchain for the community to use 😞
  - now there is...the LLVM Cascade target 😊

# Strcpy() Example

## RISC Assembly

```
strcpy:
    movi r5, 0
    bro .LBB0_1

.LBB0_1:
    add r6, r3, r5
    add r7, r4, r5
    lb r7, 0(r7)
    sb r7, 0(r6)
    addi r5, r5, 1
    tnei r6, r7, 0
    bro.t<r6> .LBB0_1
    bro.f<r6> .LBB0_2

.LBB0_2:
    ret r2
```

## EDGE Assembly

```
header
movi 0, r5 ; [0]
bro 2 ; [1]
```

```
header
read r3, n[3,0] ; [0]
read r5, n[3,1], n[2,0] ; [1]
mov n[5,1], n[9,0] ; [2]
add n[8,0] ; [3]
read r4, n[5,0] ; [4]
add n[6,0] ; [5]
lb 0, n[7,0], l[0] ; [6]
mov n[8,1], n[10,0] ; [7]
sb 0, s[1] ; [8]
addi 1, r5 ; [9]
tnei 0, b[1,p] ; [10]
bro.t b1 0 ; [11]
bro.f b1 5 ; [12]
```

```
header
read r2, n[1,0]
ret
```

# Add Immediate in TableGen



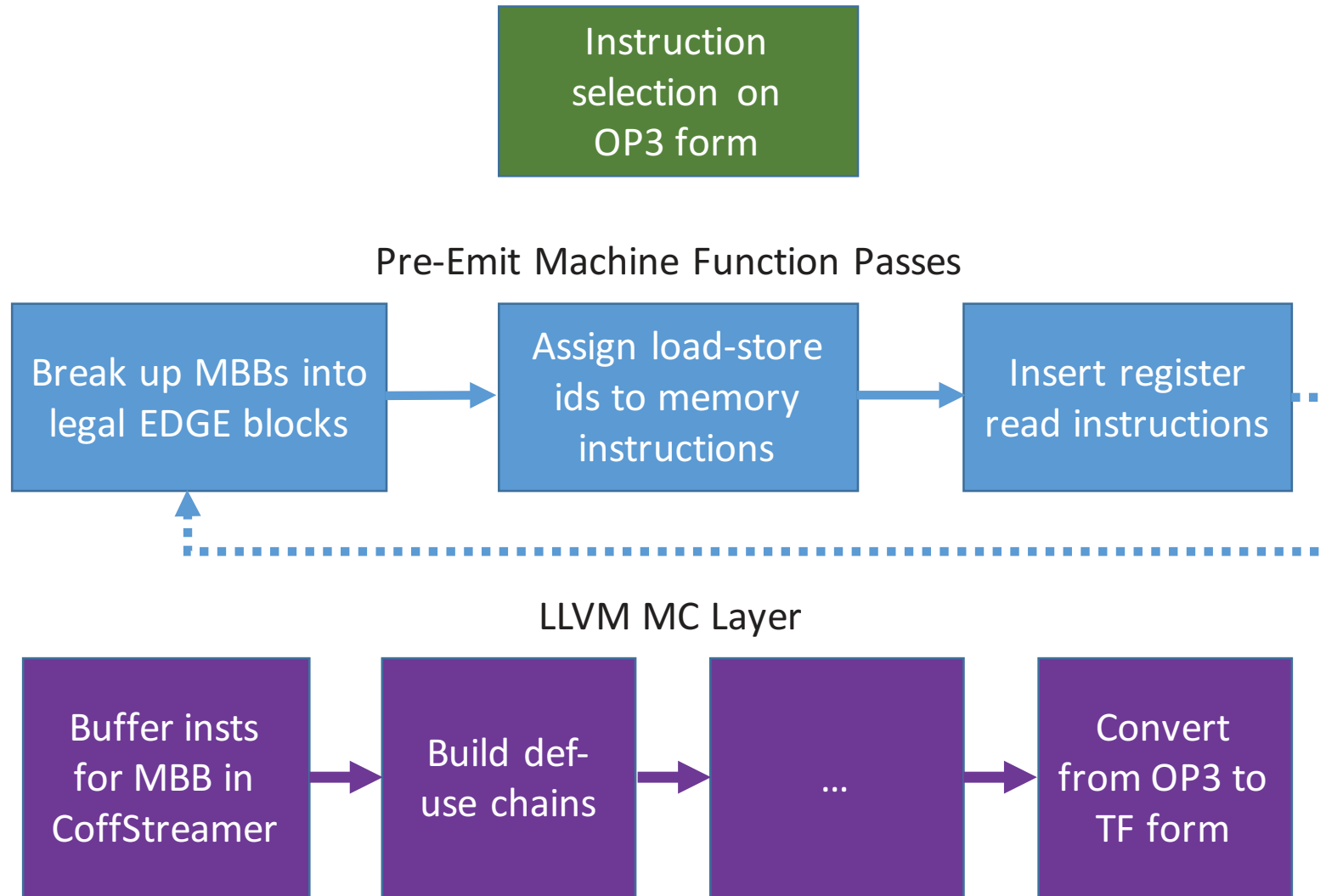
```
// Add immediate
defm ADDI : I<0x38, "addi", add>;

// Multiclass to define RISC (OP3) and EDGE (TF) forms
multiclass I<int opc, string opstr, SDPatternOperator node> {
  def _TF : I_TF<opc, opstr, []>;
  def _OP3 : I_OP3<opstr,
    [(set i64:$dst, (node i64:$src1, (i64 (immSExt9:$imm9))))]>;
}

// 1 input immediate instructions, three-address form
class I_OP3<string opstr, list<dag> pattern>
  : OP3SIMDInst<(outs GPR:$dst),
    (ins GPR:$src1, Imm90p:$imm9),
    InstFormat_I,
    opstr, "$dst, $src1, $imm9", pattern> { ... }

// Immediate instruction, 1 target form
class I_TF<int opc, string opstr, list<dag> pattern>
  : IInstr<(outs Target0p:$Target0),
    (ins Imm90p:$imm9),
    opstr, "$imm9$Target0", pattern> { ... }
```

# LLVM Backend Flow



# Visual Studio/Phoenix vs LLVM

