Projeto 3 - Apresentação

Unicamp - Universidade Estadual de Campinas
Paulo Henrique Silva Ribeiro
RA 181806
MO601

Artigo Escolhido

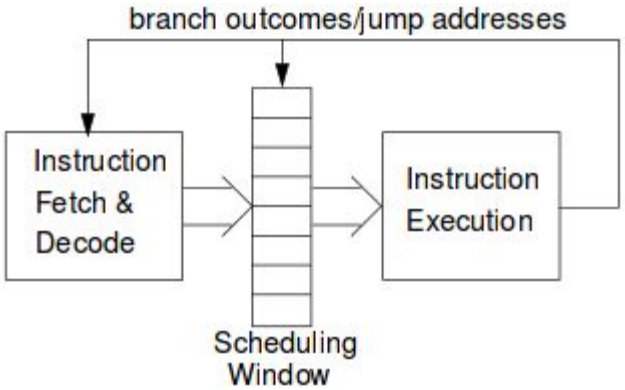
Trace Cache: a Low Latency Approach to High Bandwidth Instruction Fetching

Eric Rotenberg
Computer Science Dept.
Univ. of Wisconsin - Madison
ericro@cs.wisc.edu

Steve Bennett
Intel Corporation
sbennett@ichips.intel.com

James E. Smith
Dept. of Elec. and Comp. Engr.
Univ. of Wisconsin - Madison
jes@ece.wisc.edu

Superscalar Processors



- Especulação mais profunda;
- Mais unidades funcionais;
- IPC > 1.

Instruções não contíguas

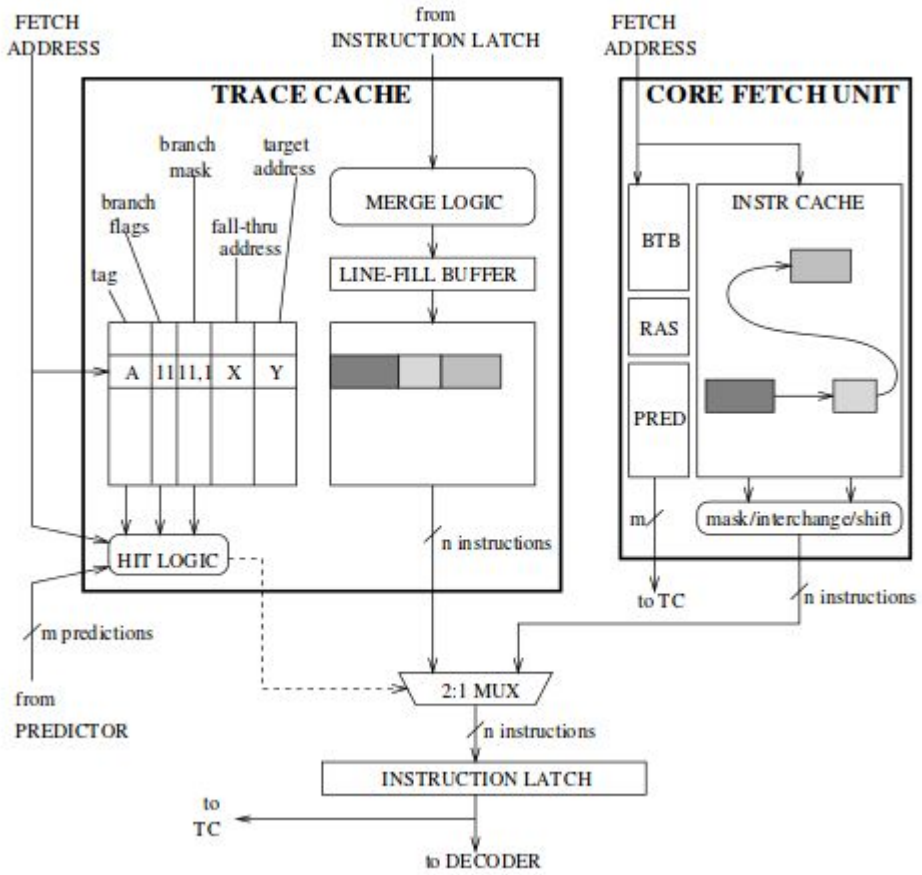
- Problema Fundamental:
  - Cache de Instrução convencional armazena na ordem de compilação

- Solução:
  - Cache de Instrução em ordem dinâmica

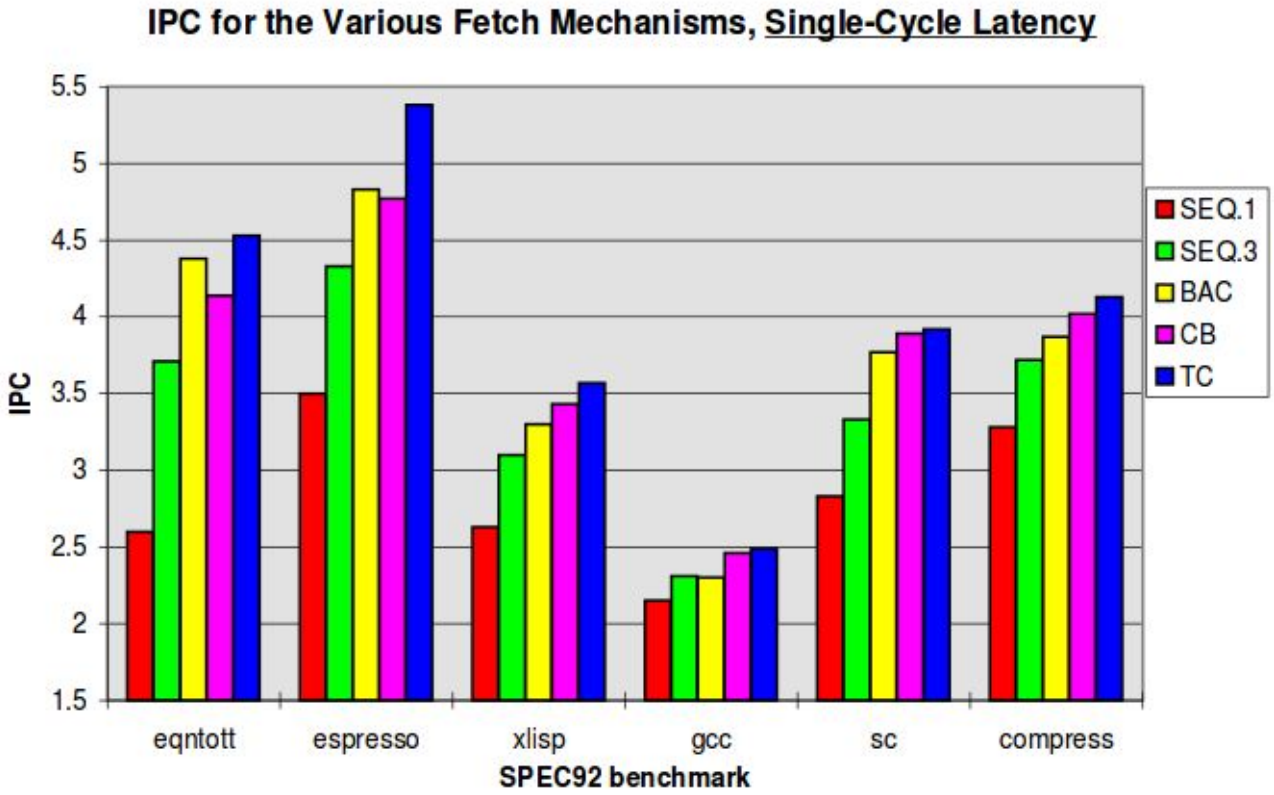
Trace Cache

- Não pretende replicar a cache de instrução convencional nem o hardware de fetch.

- New Fetch mechanism



Results


IPC for the Various Fetch Mechanisms, Single-Cycle Latency
IPC
5.5
5
4.5
4
3.5
3
2.5
2
1.5
eqntott
espresso
xlisp
gcc
sc
compress
SPEC92 benchmark
SEQ.1
SEQ.3
BAC
CB
TC

Desafios

- Desenvolver toda a nova abordagem sem suporte dos criadores;
- Desenvolver em um simulador ainda não conhecido;
- Custo para montar infra-estrutura;

Implementação

- Criar uma nova cache baseada na cache de instrução;
- Conseguir operar sobre ela;
- Detectar saída do preditor;
- Despachar instruções;

Implementação

```
MemoryManager::MemoryManager(Core* core, Network* network, ShmemPerfModel* shmem_perf_model)
   : MemoryManagerFast(core, network, shmem_perf_model)
{
   if (!dram)
      dram = new Dram(core, "dram", 150);
   if (!l3cache)
      l3cache = new CacheLocked<16, 8192>(core, "L3", MemComponent::L3_CACHE, 35, dram);
   l2cache = new Cache<8, 256>(core, "L2", MemComponent::L2_CACHE, 9, l3cache);
   icache = new Cache<4, 32>(core, "L1-I", MemComponent::L1_ICACHE, 2, l2cache);
   dcache = new Cache<8, 32>(core, "L1-D", MemComponent::L1_DCACHE, 2, l2cache);
   tcache = new Cache<4, 32>(core, "L1-T", MemComponent::L1_ICACHE, 0);
}

MemoryManager::~MemoryManager()
{
   delete icache;
   delete dcache;
   delete l2cache;
   delete tcache;
}
```

Implementação

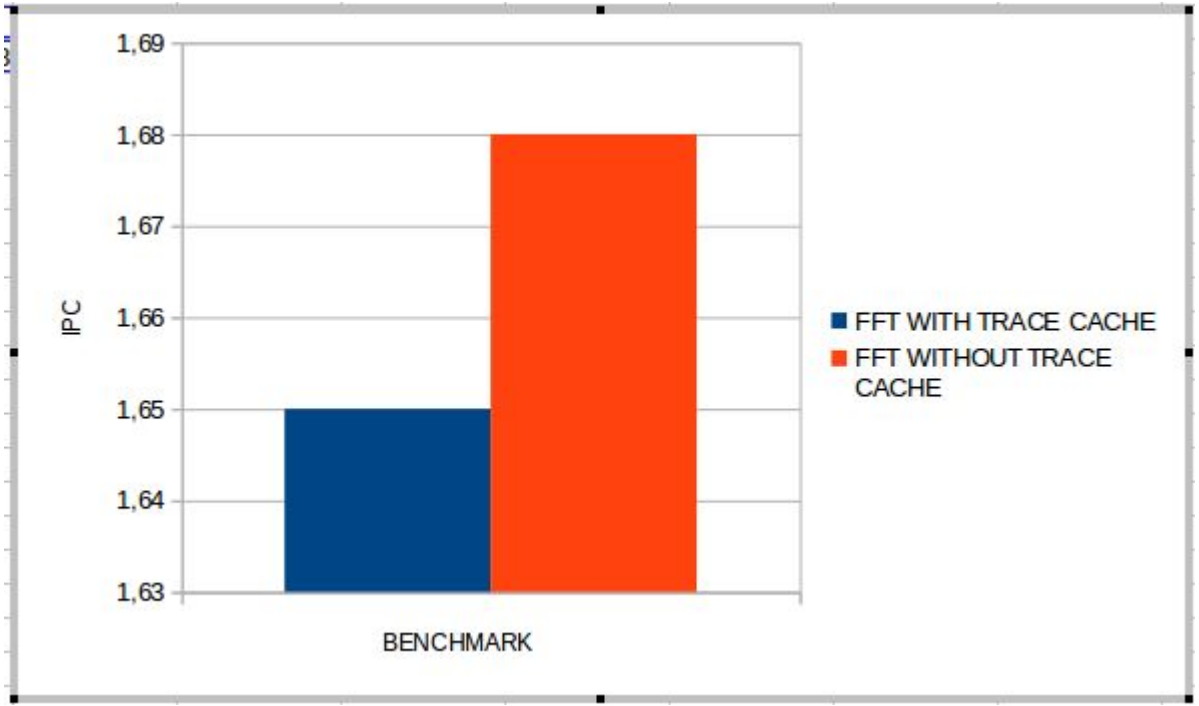
```
SubsecondTime tcache_access(Core::mem_op_t mem_op_type, IntPtr tag)
{
   if (mem_op_type == Core::WRITE) ++m_stores; else ++m_loads;
   if (m_sets[tag & m_sets_mask].find(tag))
      return SubsecondTime::Zero();
   else
   {
      if (mem_op_type == Core::WRITE) ++m_store_misses; else ++m_load_misses;
      return SubsecondTime::SECfromFloat(2);
   }
}
```

Implementação

```
SubsecondTime coreInitiateMemoryAccessFast(
      bool use_icache,
      Core::mem_op_t mem_op_type,
      IntPtr address)
{
   IntPtr tag = address >> CACHE_LINE_BITS;
   if (use_icache){
      SubsecondTime lat_tcache = tcache->tcache_access(mem_op_type, tag);
      if(lat_tcache == SubsecondTime::Zero()){
        SubsecondTime lat_write_tcache = tcache->access(Core::WRITE, tag);
        SubsecondTime lat_icache = icache->access(mem_op_type, tag) * 2;
        return lat_write_tcache + lat_icache;
      }
      else{
        return lat_tcache;
      }
   }
   else{
     return dcache->access(mem_op_type, tag);
   }

}
```

Results


1,69
1,68
1,67
1,66
1,65
1,64
1,63
IPC
BENCHMARK
FFT WITH TRACE CACHE
FFT WITHOUT TRACE CACHE

Projeto 4...

- Calcular IPC com outros tipos de preditores, para comparar com o trace cache, fazendo a análise do “High Bandwidth Instruction Fetching”.

- Simular o trace cache sobre o gcc do SPEC2006.

Conclusão