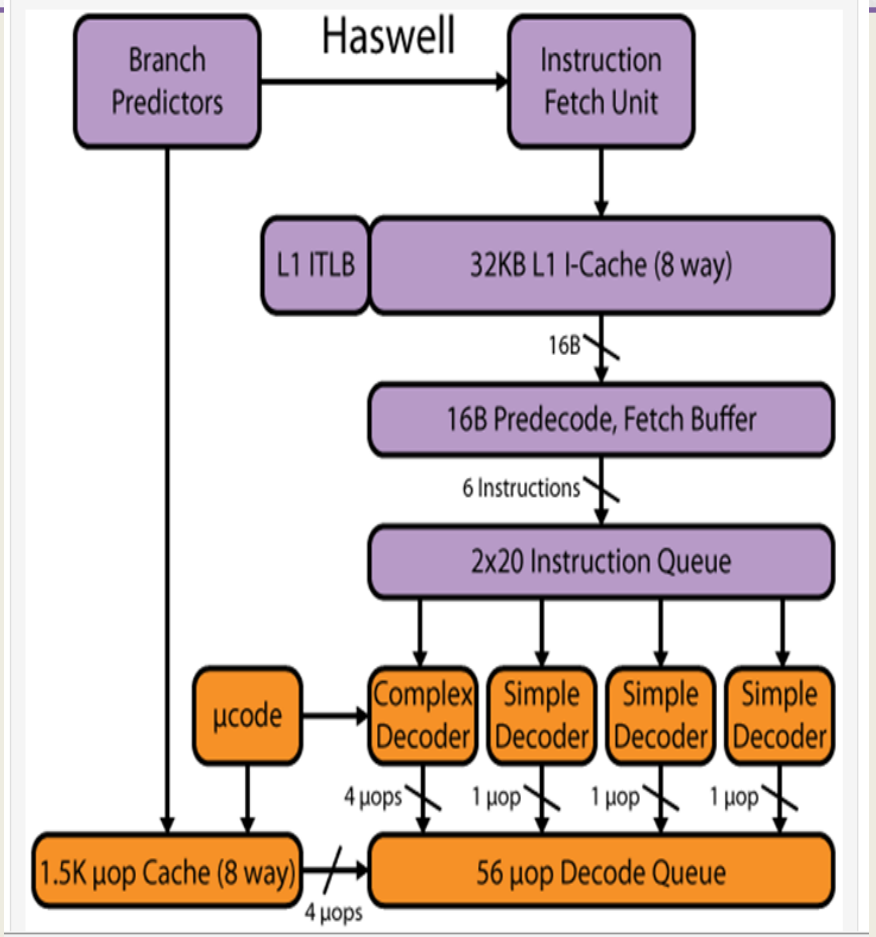
支付宝 知托付
Alipay.com

深入理解CPU的读写及优化

系统运维部—彦军

CPU前端（In order pipeline)

- 1) 分支预测
- 2）前端取指
- 3) 预解码
- 4）预解码队列
  (得到稳定的代码流）



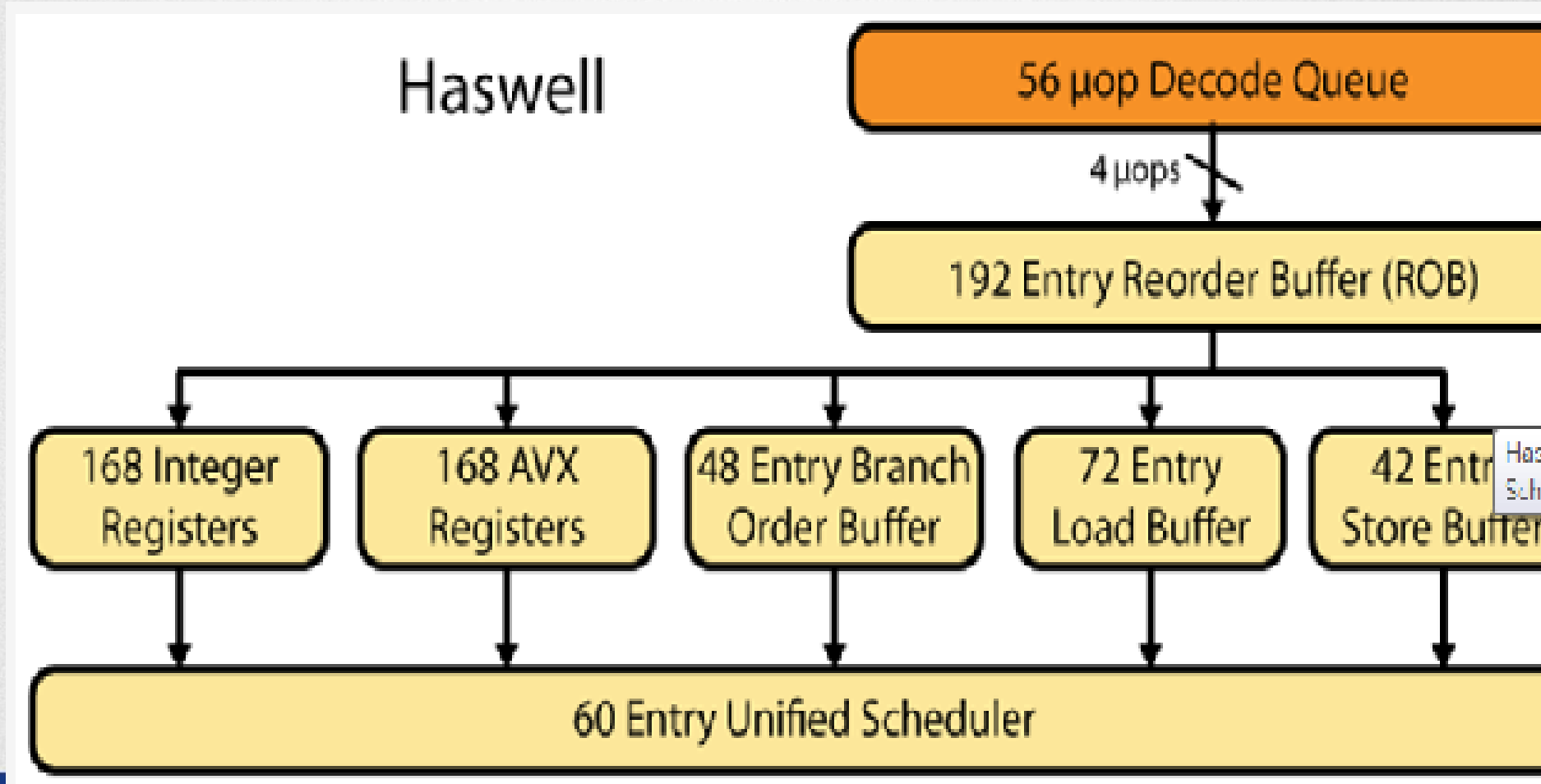
x86 CPU decode

- 4-1-1-1 rules， micro fusion & macro fusion, MS-Rom(>4 micro ops)
- Optimization rule
- a) avoid multiple micro ops if front-end or instruction cache miss or branch prediction is not bottle neck
- b) static arrange pipeline to avoid 4-4-1-1-1-1 => 4-1-1-4-1-1
- c) take full advantage of macro fusion
- d) take use of L0 cache & decode queue ASP

x86 CPU Rename & allocation

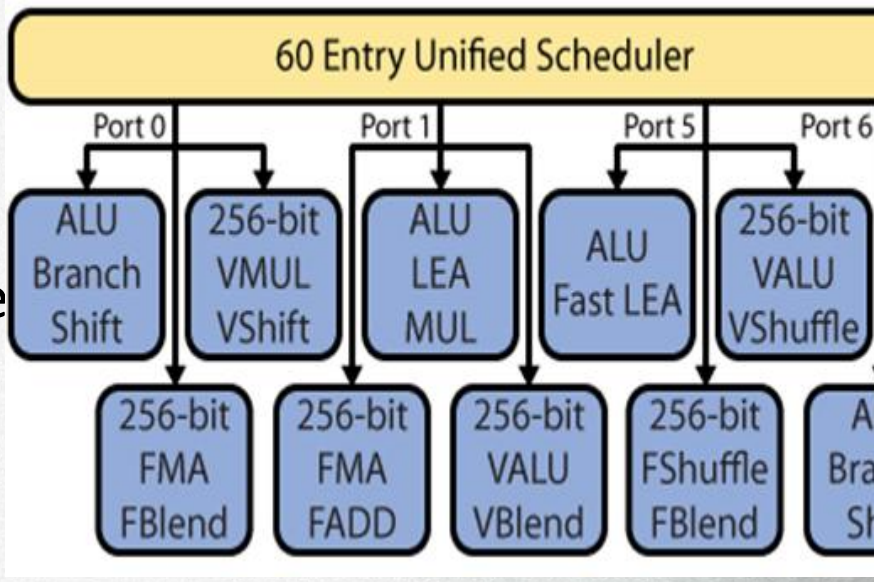
- ✓去掉　指令错误依赖(但是需要认为避免部分寄存器读写)
- ✓分配相应的资源到每条指令.
- ✓N^2 – N 强烈约束着每的周期吐出的指令条数

x86 CPU Scheduler(continue)



X86 Execution Unit

- ✓ Avoid crossing domain(integer, SIMD integer and FP (both scalar and SIMD)
- ✓ Avoid Commit conflicts
- ✓ Avoid resource pipeline compete
- ✓ Avoid memory false dependence



memory intensifies aligned cases

memcpy
内存错误依赖

拷贝32个字节

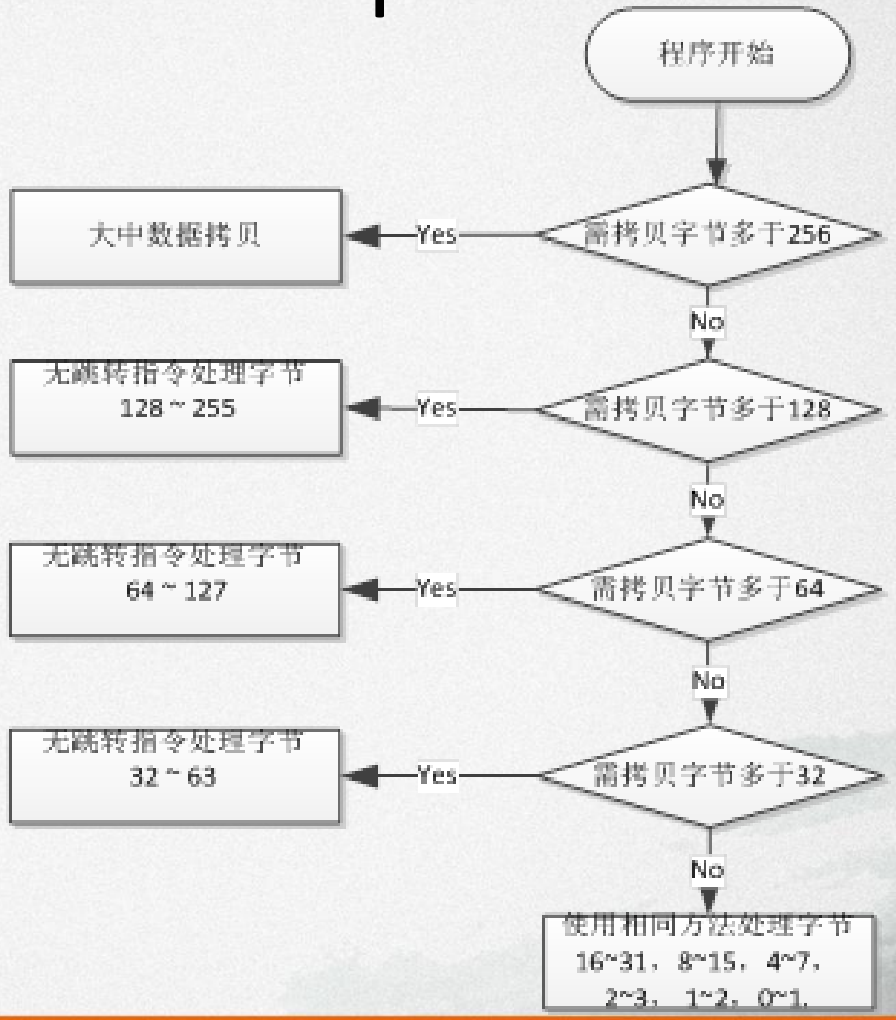
1. Movq (%rsi), %rax
2. movq %rax, (%rdi)
3. movq 8(%rsi), %rax
4. movq %rax, 8(%rdi)

1. movq 8(%rsi), %rax
2. movq %rax, 8(%rdi)
3. movq (%rsi), %rax
4. movq %rax, (%rdi)

假设rsi 为 0xf004, rdi 为 0xe008
指令 2 write 0xe008~0xe010
指令 3 read 0xf00c~ 0xf014,
如果 %rsi 和 rdi 在一个物理页面
会产生真正的依赖关系

调整之后
指令 2 write 0xe010~0xe018
指令 3 read 0xf004~0xf00c

memcpy intensifies correct branch prediction



memcpy intensifies (small size) correct branch prediction

拷贝数据从32到64个字节
*比较需拷贝字节长度是否大于
*如果需拷贝字节长度小于32则离开下面将拷贝介于32~63个字节
*拷贝源地址(rsi) 16个字节到寄存器 xmm0
*拷贝源地址（rsi +16）16个字节到寄存器 xmm1
*拷贝源地址（rsi +rdx - 32）16个字节到寄存器 xmm2
*拷贝源地址（rsi +rdx - 16）16个字节到寄存器 xmm3
*拷贝存器xmm0数据拷贝到目的地址 (rdi)
*拷贝存器xmm1数据拷贝到目的地址 (rdi + 16)
*拷贝存器xmm2数据拷贝到目的地址 (rdi + rdx - 32)
*拷贝存器xmm3数据拷贝到目的地址 (rdi + rdx - 16)
*返回

memcpy intensifies (middle size) correct branch prediction

代码如下（rsi 是源地址， rdx为需要拷贝的字节数 小于128）：
通过提前拷贝尾部128字节，当循环结束，不再需要进行任何处理，避免跳转指令的引入工作：举例说明

```
vmovups  -0x80(%rsi, %rdx), %xmm0
…
vmovups  -0x10(%rsi, %rdx), %xmm7
Vmovups %xmm0, -0x80(%rdi, %rdx)
…
Vmovups %xmm7, -0x10(%rdi, %rdx)

Loop:
Copy 128 byte

Sub  0x80, %rdx
Jae L(Loop)
ret
```

memcpy intensifies (large size) correct branch prediction

1) Rep movsb /*intel fast instruction */
2) non-temporary instruction

随机测试表明 跳转指令错误率减少 20%， CPU2006 403.gcc 告诉我们性能提升 24%，

3）根据我们的测试和优化，二篇优化跳转预测的专利将会在本月提交,其中一篇已经录入