Modern Dynamic Instruction Scheduling
Tomasulo Performance

Observe at the EX stage, how many cycles to execute this code?

LW  R2, 45 (R3)
ADD  R6, R2, R4
SUB  R10, R0, R6
ADD  R10, R10, R12

Assume load takes 1 cycle, ALU 1 cycle
Tomasulo vs MIPS Pipeline

How many cycles on the 5-stage MIPS pipeline?

Why does the simple pipeline run faster?
Review Tomasulo Inst Scheduling

Both in RS, no contention on CDB or FU

ADD R2, R2, 45  # R2=>tag p, result = A
SUB R6, R2, R4  # R4 is ready, = B

Cycle 1:       ADD starts at FU, producing A
Cycle 2:       ADD broadcast p + A
                SUB matches on p and accepts A
Cycle 3:       SUB starts execution, FU calc A-B

A is produced at cycle 1, but consumed at cycle 3 -- unavoidable?
Review Data Forwarding

MIPS pipeline data forwarding:
FU/MEM => FU
Why not in Tomasulo?

But tag broadcasting has one cycle delay!!

When is it known that A will be ready?

Cycle 1: A is to be ready
Cycle 2: A and its tag are broadcast

If tag is broadcast one-cycle earlier ...

Cycle 2: forward A from FU output to FU input...
Revise Scheduling

RS1: ADD R6, R2, R4
RS2: SUB R10, R0, R6
RS3: ADD R12, R10, R6

ADD(1) has been ready and selected
1. - ADD(1)'s tag is broadcast, and operands are sent to FU;
   - SUB is waken up and selected;
2. - SUB's tag is broadcast, operands are sent to FU;
   - forwarding logic replace 2nd FU operand with FU output;
   - ADD(2) is waken up and accepts FU output, and is selected
3. So on and so forth...

RS can be centralized or distributed

--- One cycle earlier

How to address CDB contention?
How to Handle Variable Latency?

One method: Use result shift register to track latency and control tag/data bus
Revised Pipeline Stages

- As efficient as MIPS pipeline (instruction throughput)
- With data forwarding and bypassing
Rethink RS and ROB design

Data broadcasting to RS stations:
- Broadcasting saves reg-write to reg-read delay
- \( n \) child instructions can receive data simultaneously

However,
- RS and ROB may store duplicate values
- Not all \( n \) child instructions may fu-execute next cycle
Rethink ROB Design

Does every ROB entry store register output?
Split ROB into ROB + Rename Register

Arch. Regs → Rename Regs → ROB (no reg value)
From FU
Alternative: Register Mapping and Issue Queue

RS entry

<table>
<thead>
<tr>
<th>op</th>
<th>Qj</th>
<th>Qk</th>
<th>busy</th>
<th>Vj</th>
<th>Vk</th>
</tr>
</thead>
</table>

ROB entry

<table>
<thead>
<tr>
<th>i-type</th>
<th>dest</th>
<th>PC</th>
<th>valid</th>
<th>result</th>
</tr>
</thead>
</table>

Arch. regs

Physical register

| p1 | p2 | p3 | p_n |

Physical register: Central collection of all register values, architectural or temporary
Register Mapping Table

- Rename architectural register to physical register
- NO real architectural registers (now virtual register)
- RS => issue queue
- Rename stage: allocate issue queue entry, allocate ROB, allocate physical register
- What is tag now?
Mis-speculation Recovery

- RS+ROB: no changes to arch. registers, so just clear pipeline and re-fetch
- Fundamental issue: software does not see wrong register contents

Recovery for mapping approach: Roll back mapping table to the mis-speculation point

Architectural registers
=> virtual registers

How to implement mapping table supporting recovery?
Scheduling with Issue Queue

- FETCH
- RENAME
- SCHEDULE
- REG
- EXE/WB
- COMMIT

Diagram:
- IM
  - Fetch Unit
    - Decode
    - Rename
    - Issue Queue
      - phy. regfile
        - S-buf
        - L-buf
        - FU1
        - FU2
        - DM
    - ROB
Alpha 21264 Pipeline
Examples: Intel P6

... 

Fetch 
Decode 
Rename 
ROB Rd 

... 

Use RS + ROB

- 40-entry ROB
- 20-entry RS station
- Register Alias Table
Example: Intel Pentium 4

Use issue queue + phy. regs
Generic Superscalar Processor Models

Issue queue based

Reservation based

Source: Paracharla PhD thesis 1998
Dynamic Scheduling in P6
(Pentium Pro, II, III)

Q: How pipeline 1 to 17 byte 80x86 instructions?

◆ P6 doesn’t pipeline 80x86 instructions
◆ P6 decode unit translates the Intel instructions into 72-bit micro-operations (~ MIPS)
◆ Sends micro-operations to reorder buffer & reservation stations
◆ Many instructions translate to 1 to 4 micro-operations
◆ Complex 80x86 instructions are executed by a conventional microprogram (8K x 72 bits) that issues long sequences of micro-operations
◆ 14 clocks in total pipeline (~ 3 state machines)
Dynamic Scheduling in P6

<table>
<thead>
<tr>
<th>Parameter</th>
<th>80x86 microops</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. instructions issued/clock</td>
<td>3</td>
</tr>
<tr>
<td>Max. instr. complete exec./clock</td>
<td>5</td>
</tr>
<tr>
<td>Max. instr. commited/clock</td>
<td>3</td>
</tr>
<tr>
<td>Window (Instrs in reorder buffer)</td>
<td>40</td>
</tr>
<tr>
<td>Number of reservations stations</td>
<td>20</td>
</tr>
<tr>
<td>Number of rename registers</td>
<td>40</td>
</tr>
<tr>
<td>No. integer functional units (FUs)</td>
<td>2</td>
</tr>
<tr>
<td>No. floating point FUs</td>
<td>1</td>
</tr>
<tr>
<td>No. SIMD Fl. Pt. FUs</td>
<td>1</td>
</tr>
<tr>
<td>No. memory Fus</td>
<td>1 load + 1 store</td>
</tr>
</tbody>
</table>
P6 Pipeline

- 14 clocks in total (~3 state machines)
- 8 stages are used for in-order instruction fetch, decode, and issue
  - Takes 1 clock cycle to determine length of 80x86 instructions + 2 more to create the micro-operations (uops)
- 3 stages are used for out-of-order execution in one of 5 separate functional units
- 3 stages are used for instruction commit
P6 Block Diagram
Pentium III Die Photo

- EBL/BBL - Bus logic, Front, Back
- MOB - Memory Order Buffer
- Packed FPU - MMX Fl. Pt. (SSE)
- IEU - Integer Execution Unit
- FAU - Fl. Pt. Arithmetic Unit
- MIU - Memory Interface Unit
- DCU - Data Cache Unit
- PMH - Page Miss Handler
- DTLB - Data TLB
- BAC - Branch Address Calculator
- RAT - Register Alias Table
- SIMD - Packed Fl. Pt.
- RS - Reservation Station
- BTB - Branch Target Buffer
- IFU - Instruction Fetch Unit (+I$)
- ID - Instruction Decode
- ROB - Reorder Buffer
- MS - Micro-instruction Sequence

1st Pentium III, Katmai: 9.5 M transistors, 12.3 * 10.4 mm in 0.25-mi. with 5 layers of aluminum
AMD Althon

- Similar to P6 microarchitecture (Pentium III), but more resources
- Transistors: PIII 24M v. Althon 37M
- Die Size: 106 mm$^2$ v. 117 mm$^2$
- Power: 30W v. 76W
- Cache: 16K/16K/256K v. 64K/64K/256K
- Window size: 40 vs. 72 uops
- Rename registers: 40 v. 36 int +36 Fl. Pt.
- BTB: 512 x 2 v. 4096 x 2
- Pipeline: 10-12 stages v. 9-11 stages
- Clock rate: 1.0 GHz v. 1.2 GHz
- Memory bandwidth: 1.06 GB/s v. 2.12 GB/s
Pentium 4

- Still translate from 80x86 to micro-ops
- P4 has better branch predictor, more FUs
- Instruction Cache holds micro-operations vs. 80x86 instructions
  - no decode stages of 80x86 on cache hit
  - called “trace cache” (TC)
- Faster memory bus: 400 MHz v. 133 MHz
- Caches
  - Pentium III: L1I 16KB, L1D 16KB, L2 256 KB
  - Pentium 4: L1I 12K uops, L1D 8 KB, L2 256 KB
  - Block size: PIII 32B v. P4 128B; 128 v. 256 bits/clock
- Clock rates:
  - Pentium III 1 GHz v. Pentium IV 1.5 GHz
Pentium 4 features

- Multimedia instructions 128 bits wide vs. 64 bits wide
  => 144 new instructions
  - When used by programs?
  - Faster Floating Point: execute 2 64-bit FP Per clock
  - Memory FU: 1 128-bit load, 1 128-store /clock to MMX regs
- Using RAMBUS DRAM
  - Bandwidth faster, latency same as SDRAM
  - Cost 2X-3X vs. SDRAM
- ALUs operate at 2X clock rate for many ops
- Pipeline doesn’t stall at this clock rate: uops replay
- Rename registers: 40 vs. 128; Window: 40 v. 126
- BTB: 512 vs. 4096 entries (Intel: 1/3 improvement)
## Basic Pentium 4 Pipeline

<table>
<thead>
<tr>
<th>TC Nxt IP</th>
<th>TC Fetch</th>
<th>Drive</th>
<th>Alloc</th>
<th>Rename</th>
<th>Queue</th>
<th>Schd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schd</td>
<td>Schd</td>
<td>Disp</td>
<td>Disp</td>
<td>Reg</td>
<td>Ex</td>
<td>Flags</td>
</tr>
</tbody>
</table>

1-2 trace cache next instruction pointer
3-4 fetch uops from Trace Cache
5 drive upos to alloc
6 alloc resources (ROB, reg, ...)
7-8 rename arch. reg to 128 physical reg
9 put renamed uops into queue
10-12 write uops into scheduler
13-14 move up to 6 uops to FU
15-16 read registers
17 FU execution
18 computer flags e.g. for branch instructions
19 check branch output with branch prediction
20 drive branch check result to frontend
Block Diagram of Pentium 4 Microarchitecture

- BTB = Branch Target Buffer (branch predictor)
- I-TLB = Instruction TLB, Trace Cache = Instruction cache
- RF = Register File; AGU = Address Generation Unit
- "Double pumped ALU" means ALU clock rate 2X => 2X ALU F.U.s

From "Pentium 4 (Partially) Previewed," Microprocessor Report, 8/28/00
Pentium 4 Die Photo

- 42M Xtors
  - PIII: 26M
- 217 \text{mm}^2
  - PIII: 106 \text{mm}^2
- L1 Execution Cache
  - Buffer 12,000 Micro-Ops
- 8KB data cache
- 256KB L2$
### Workstation Microprocessors

3/2001

<table>
<thead>
<tr>
<th>Processor</th>
<th>Alpha 21264B</th>
<th>AMD Athlon</th>
<th>HP PA-8600</th>
<th>IBM Power3-II</th>
<th>Intel Pentium III</th>
<th>Intel Pentium 4</th>
<th>MIPS R12000</th>
<th>Sun Ultra-II</th>
<th>Sun Ultra-III</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Rate</td>
<td>833MHz</td>
<td>1.2GHz</td>
<td>552MHz</td>
<td>450MHz</td>
<td>1.0GHz</td>
<td>1.5GHz</td>
<td>400MHz</td>
<td>480MHz</td>
<td>900MHz</td>
</tr>
<tr>
<td>Cache (I/D/L2)</td>
<td>64K/64K</td>
<td>64K/64K/256K</td>
<td>512K/1M</td>
<td>32K/64K</td>
<td>16K/16K/256K</td>
<td>12K/8K/256K</td>
<td>32K/32K</td>
<td>16K/16K</td>
<td>32K/64K</td>
</tr>
<tr>
<td>Issue Rate</td>
<td>4 issue</td>
<td>3 x 86 instr</td>
<td>4 issue</td>
<td>4 issue</td>
<td>3 x 86 instr</td>
<td>3 x ROPs</td>
<td>4 issue</td>
<td>4 issue</td>
<td>4 issue</td>
</tr>
<tr>
<td>Pipeline Stages</td>
<td>7/9 stages</td>
<td>9/11 stages</td>
<td>7/9 stages</td>
<td>7/8 stages</td>
<td>12/14 stages</td>
<td>22/24 stages</td>
<td>6 stages</td>
<td>6/9 stages</td>
<td>14/15 stages</td>
</tr>
<tr>
<td>Out of Order</td>
<td>80 instr</td>
<td>72ROPs</td>
<td>56 instr</td>
<td>32 instr</td>
<td>40 ROPs</td>
<td>126 ROPs</td>
<td>48 instr</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Rename regs</td>
<td>48/41</td>
<td>36/36</td>
<td>56 total</td>
<td>16 int/24 fp</td>
<td>40 total</td>
<td>128 total</td>
<td>32/32</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>BHT Entries</td>
<td>4K x 9-bit</td>
<td>4K x 2-bit</td>
<td>2K x 2-bit</td>
<td>2K x 2-bit</td>
<td>&gt;= 512</td>
<td>4K x 2-bit</td>
<td>2K x 2-bit</td>
<td>512 x 2-bit</td>
<td>16K x 2-bit</td>
</tr>
<tr>
<td>TLB Entries</td>
<td>128/128</td>
<td>280/288</td>
<td>120 unified</td>
<td>128/128</td>
<td>32I / 64D</td>
<td>128I/65D</td>
<td>64 unified</td>
<td>64I/64D</td>
<td>128I/512D</td>
</tr>
<tr>
<td>Memory B/W</td>
<td>2.66GB/s</td>
<td>2.1GB/s</td>
<td>1.54GB/s</td>
<td>1.6GB/s</td>
<td>1.06GB/s</td>
<td>3.2GB/s</td>
<td>539 MB/s</td>
<td>1.9GB/s</td>
<td>4.8GB/s</td>
</tr>
<tr>
<td>Package</td>
<td>CPGA-588</td>
<td>PGA-462</td>
<td>LGA-544</td>
<td>SCC-1088</td>
<td>PGA-370</td>
<td>PGA-423</td>
<td>CPGA-527</td>
<td>CLGA-787</td>
<td>1368 FC-LGA</td>
</tr>
<tr>
<td>IC Process</td>
<td>0.18μ 6M</td>
<td>0.18μ 6M</td>
<td>0.25μ 2M</td>
<td>0.22μ 6M</td>
<td>0.18μ 6M</td>
<td>0.18μ 6M</td>
<td>0.25μ 4M</td>
<td>0.29μ 6M</td>
<td>0.18μ 7M</td>
</tr>
<tr>
<td>Die Size</td>
<td>115mm²</td>
<td>117mm²</td>
<td>477mm²</td>
<td>163mm²</td>
<td>106mm²</td>
<td>217mm²</td>
<td>204mm²</td>
<td>126 mm²</td>
<td>210mm²</td>
</tr>
<tr>
<td>Transistors</td>
<td>15.4 million</td>
<td>37 million</td>
<td>130 million</td>
<td>23 million</td>
<td>24 million</td>
<td>42 million</td>
<td>7.2 million</td>
<td>3.8 million</td>
<td>29 million</td>
</tr>
<tr>
<td>Est mfg cost*</td>
<td>$160</td>
<td>$62</td>
<td>$330</td>
<td>$110</td>
<td>$39</td>
<td>$110</td>
<td>$125</td>
<td>$70</td>
<td>$145</td>
</tr>
<tr>
<td>Power(Max)</td>
<td>75W*</td>
<td>76W</td>
<td>60W*</td>
<td>36W*</td>
<td>30W</td>
<td>55W(TDP)</td>
<td>25W*</td>
<td>20W*</td>
<td>65W</td>
</tr>
<tr>
<td>Availability</td>
<td>1Q01</td>
<td>4Q00</td>
<td>3Q00</td>
<td>4Q00</td>
<td>2Q00</td>
<td>4Q00</td>
<td>2Q00</td>
<td>3Q00</td>
<td>4Q00</td>
</tr>
</tbody>
</table>

- **Max issue**: 4 instructions (many CPUs)
- **Max rename registers**: 128 (Pentium 4)
- **Max BHT**: 4K x 9 (Alpha 21264B), 16Kx2 (Ultra III)
- **Max Window Size (OOO)**: 126 instructions (Pent. 4)
- **Max Pipeline**: 22/24 stages (Pentium 4)

*Source: Microprocessor Report, www.MPRonline.com*
<table>
<thead>
<tr>
<th>Processor</th>
<th>Alpha 21264B</th>
<th>AMD Athlon</th>
<th>HP PA-8600</th>
<th>IBM Power 3-II</th>
<th>Intel PIII</th>
<th>Intel P4</th>
<th>MIPS R12000</th>
<th>Sun Ultra-II</th>
<th>Sun Ultra-III</th>
</tr>
</thead>
<tbody>
<tr>
<td>System or Motherboard</td>
<td>Alpha ES40 Model 6</td>
<td>AMD GA-7ZM</td>
<td>HP9000 j6000</td>
<td>RS/6000 44P-170</td>
<td>Dell Prec. 420</td>
<td>Intel P5 850/18</td>
<td>SGI 2200</td>
<td>Sun Enterps 450</td>
<td>Sun Blade 1000</td>
</tr>
<tr>
<td>Clock Rate</td>
<td>833MHz</td>
<td>1.2GHz</td>
<td>552MHz</td>
<td>450MHz</td>
<td>1GHz</td>
<td>1.5GHz</td>
<td>1.8GHz</td>
<td>400MHz</td>
<td>480MHz</td>
</tr>
<tr>
<td>External Cache</td>
<td>8MB</td>
<td>None</td>
<td>None</td>
<td>8MB</td>
<td>None</td>
<td>None</td>
<td>8MB</td>
<td>8MB</td>
<td>8MB</td>
</tr>
<tr>
<td>164.gzip</td>
<td>392</td>
<td>n/a</td>
<td>376</td>
<td>230</td>
<td>545</td>
<td>553</td>
<td>226</td>
<td>165</td>
<td>349</td>
</tr>
<tr>
<td>175.vpr</td>
<td>452</td>
<td>n/a</td>
<td>421</td>
<td>285</td>
<td>354</td>
<td>298</td>
<td>384</td>
<td>212</td>
<td>383</td>
</tr>
<tr>
<td>176.gcc</td>
<td>617</td>
<td>n/a</td>
<td>577</td>
<td>350</td>
<td>401</td>
<td>588</td>
<td>313</td>
<td>232</td>
<td>500</td>
</tr>
<tr>
<td>181.mcf</td>
<td>441</td>
<td>n/a</td>
<td>384</td>
<td>498</td>
<td>276</td>
<td>473</td>
<td>563</td>
<td>356</td>
<td>474</td>
</tr>
<tr>
<td>186.crafty</td>
<td>694</td>
<td>n/a</td>
<td>472</td>
<td>304</td>
<td>523</td>
<td>497</td>
<td>334</td>
<td>175</td>
<td>439</td>
</tr>
<tr>
<td>197.parser</td>
<td>360</td>
<td>n/a</td>
<td>361</td>
<td>171</td>
<td>362</td>
<td>472</td>
<td>283</td>
<td>211</td>
<td>412</td>
</tr>
<tr>
<td>252.eon</td>
<td>645</td>
<td>n/a</td>
<td>395</td>
<td>280</td>
<td>615</td>
<td>650</td>
<td>360</td>
<td>209</td>
<td>465</td>
</tr>
<tr>
<td>253.perlbmk</td>
<td>526</td>
<td>n/a</td>
<td>406</td>
<td>215</td>
<td>614</td>
<td>703</td>
<td>246</td>
<td>247</td>
<td>457</td>
</tr>
<tr>
<td>254.gap</td>
<td>365</td>
<td>n/a</td>
<td>229</td>
<td>256</td>
<td>443</td>
<td>708</td>
<td>204</td>
<td>171</td>
<td>300</td>
</tr>
<tr>
<td>255.vortex</td>
<td>673</td>
<td>n/a</td>
<td>764</td>
<td>312</td>
<td>717</td>
<td>735</td>
<td>294</td>
<td>304</td>
<td>581</td>
</tr>
<tr>
<td>256.bzip2</td>
<td>560</td>
<td>n/a</td>
<td>349</td>
<td>258</td>
<td>396</td>
<td>420</td>
<td>334</td>
<td>237</td>
<td>500</td>
</tr>
<tr>
<td>300.twolf</td>
<td>658</td>
<td>n/a</td>
<td>479</td>
<td>414</td>
<td>394</td>
<td>403</td>
<td>451</td>
<td>243</td>
<td>473</td>
</tr>
<tr>
<td>SPECint_base2000</td>
<td>518</td>
<td>n/a</td>
<td>417</td>
<td>286</td>
<td>454</td>
<td>524</td>
<td>320</td>
<td>225</td>
<td>438</td>
</tr>
<tr>
<td>168.wupside</td>
<td>529</td>
<td>360</td>
<td>340</td>
<td>360</td>
<td>416</td>
<td>759</td>
<td>280</td>
<td>284</td>
<td>497</td>
</tr>
<tr>
<td>171.swim</td>
<td>1,156</td>
<td>506</td>
<td>761</td>
<td>279</td>
<td>493</td>
<td>1,244</td>
<td>300</td>
<td>285</td>
<td>752</td>
</tr>
<tr>
<td>172.mgrid</td>
<td>580</td>
<td>272</td>
<td>462</td>
<td>319</td>
<td>274</td>
<td>558</td>
<td>231</td>
<td>226</td>
<td>377</td>
</tr>
<tr>
<td>173.applu</td>
<td>424</td>
<td>298</td>
<td>563</td>
<td>327</td>
<td>280</td>
<td>641</td>
<td>237</td>
<td>150</td>
<td>221</td>
</tr>
<tr>
<td>177.mesa</td>
<td>713</td>
<td>302</td>
<td>300</td>
<td>330</td>
<td>541</td>
<td>553</td>
<td>289</td>
<td>273</td>
<td>469</td>
</tr>
<tr>
<td>178.galgel</td>
<td>558</td>
<td>468</td>
<td>569</td>
<td>429</td>
<td>335</td>
<td>537</td>
<td>989</td>
<td>735</td>
<td>1,266</td>
</tr>
<tr>
<td>179.art</td>
<td>1,540</td>
<td>213</td>
<td>419</td>
<td>969</td>
<td>410</td>
<td>514</td>
<td>995</td>
<td>920</td>
<td>990</td>
</tr>
<tr>
<td>183.equake</td>
<td>231</td>
<td>236</td>
<td>347</td>
<td>560</td>
<td>249</td>
<td>739</td>
<td>222</td>
<td>149</td>
<td>211</td>
</tr>
<tr>
<td>187.facerec</td>
<td>822</td>
<td>411</td>
<td>258</td>
<td>257</td>
<td>307</td>
<td>451</td>
<td>411</td>
<td>459</td>
<td>718</td>
</tr>
<tr>
<td>188.ammp</td>
<td>488</td>
<td>221</td>
<td>376</td>
<td>326</td>
<td>294</td>
<td>366</td>
<td>373</td>
<td>313</td>
<td>421</td>
</tr>
<tr>
<td>189.lucas</td>
<td>731</td>
<td>237</td>
<td>370</td>
<td>284</td>
<td>349</td>
<td>764</td>
<td>259</td>
<td>205</td>
<td>204</td>
</tr>
<tr>
<td>191.fma3d</td>
<td>528</td>
<td>365</td>
<td>302</td>
<td>340</td>
<td>297</td>
<td>427</td>
<td>192</td>
<td>207</td>
<td>302</td>
</tr>
<tr>
<td>200.sixtrack</td>
<td>340</td>
<td>256</td>
<td>286</td>
<td>234</td>
<td>170</td>
<td>257</td>
<td>199</td>
<td>159</td>
<td>273</td>
</tr>
<tr>
<td>301.aspi</td>
<td>553</td>
<td>278</td>
<td>523</td>
<td>349</td>
<td>371</td>
<td>427</td>
<td>252</td>
<td>189</td>
<td>340</td>
</tr>
<tr>
<td>SPECfp_base2000</td>
<td>590</td>
<td>304</td>
<td>400</td>
<td>356</td>
<td>329</td>
<td>549</td>
<td>319</td>
<td>274</td>
<td>427</td>
</tr>
</tbody>
</table>
Benchmarks: Pentium 4 v. PIII v. Althon

✔ SPECbase2000
  ■ Int, P4@1.5 GHz: 524, PIII@1GHz: 454, AMD Althon@1.2Ghz:?
  ■ FP, P4@1.5 GHz: 549, PIII@1GHz: 329, AMD Althon@1.2Ghz:304

✔ WorldBench 2000 benchmark (business) PC World magazine, Nov. 20, 2000 (bigger is better)
  ■ P4 : 164, PIII : 167, AMD Althon: 180

✔ Quake 3 Arena: P4 172, Althon 151

✔ SYSmark 2000 composite: P4 209, Althon 221

✔ Office productivity: P4 197, Althon 209

✔ S.F. Chronicle 11/20/00: "... the challenge for AMD now will be to argue that frequency is not the most important thing-- precisely the position Intel has argued while its Pentium III lagged behind the Athlon in clock speed."
Summary of Dynamic Scheduling

◆ Pipeline stages
  ■ Renaming (in-order)
  ■ Schedule
  ■ Commit (in-order)

◆ Two organizations
  ■ Mapping table + phy reg + issue queue + ROB;
    \( REN \rightarrow SCHD \rightarrow REG \)
  ■ Reg alias table + RS + ROB, reg in RS and ROB;
    \( REN \rightarrow REG \rightarrow SCHD \)

◆ Scheduling methods
  ■ Tag broadcasting vs. scoreboard (later)

◆ CDC6600: introduces scoreboard

◆ Tomasulo: introduces renaming and tag broadcasting

◆ Reorder buffer: provides in-order commit

◆ Real OOO processors
  ■ very complicated (like a vehicle)
  ■ bring impl variants
  ■ but all root in those basic designs