Fundusze Europejskie
Wiedza Edukacja Rozwój

Rzeczpospolita Polska

Unia Europejska
Europejski Fundusz Społeczny

Politechnika Śląska jako Centrum Nowoczesnego Kształcenia
opartego o badania i innowacje

POWR.03.05.00-IP.08-00-PZ1/17

Projekt współfinansowany przez Unię Europejską ze środków Europejskiego Funduszu Społecznego

Microprocessor and Embedded Systems

Faculty of Automatic Control, Electronics and Computer Science,
Informatics, Bachelor Degree

Lecture 9

Microprocessor operation acceleration

Bartłomiej Zieliński, PhD, DSc

µp acceleration

Program:

- Acceleration/optimisation?
- Pipelining
- Superscalar processing
- Branch prediction
- Cache memory

µp acceleration

- Acceleration/optimisation?
  - Clock frequency up
    - Frequency limit in silicon structure
  - Forms of parallel execution
    - Pipelining
    - Superscalar
    - Full parallel
  - RISC/CISC/FISC

µp acceleration

- Typical RISC properties
  - Constant instruction format
    - Easier decoding
  - Memory only for rd/wr operations
  - Many registers (32+)
    - No outlined accumulator
    - Argument passing through registers (not stack)
  - Small number of command
  - Fast command execution (1 clk/cmd)
  - Simple decoding/control unit
  - (Harvard achitecture)

µp acceleration

- Pipeline execution
  - Command split into phases
    - E.g., Fetch/Decode/Address/Exec/Write
    - *Optimum≈8 phases?*
  - Few commands processed concurrently
    - Each in another processing phase
  - Problems
    - Command time in different stages varies → queues
    - Exec interruption → pipe emptied → big delay
    - Command & data mutual dependencies

µp acceleration

- Superscalar execution
  - More than 1 execution pipe
    - >>1 not effective due to dependencies
    - Pipes may have different capabilities
  - Command sequence split into pipes
    - Depends on differences between pipes
    - Sychronous/asynchronous pipes
      - Syn: pipe1 waits → pipe2 waits too even if no reason
      - Asyn: pipe1 waits → pipe2 goes on
        - » Cmd2 ends before cmd1
          - ***Out-of-order execution***

µp acceleration

- Operand dependencies
  - *Read after read*
    - E.g., B=B+C || A=C

      → ***Dual Pipe Access***
  - *Read after Write*
    - E.g., A=A+B || C=A; A=A+B || [M]=A

      → ***Result forwarding**, **Operand forwarding***
  - *Write after Read*
    - E.g., B=A || A=A+C

      → ***Register renaming***
  - *Write after Write*
    - E.g., A=[M] || A=A+B

µp acceleration

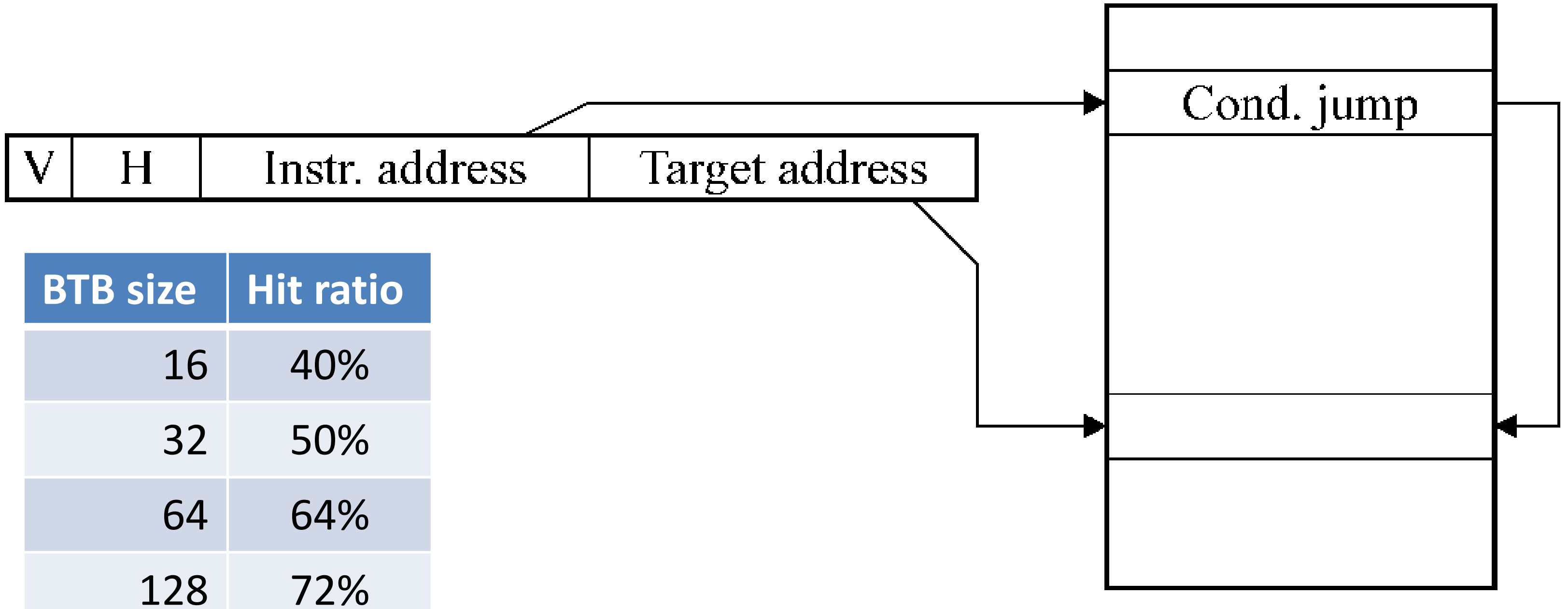
- Branch prediction
  - 3 kinds of code execution disturbances:
    - Interrupts
    - Unconditional jumps
    - Conditional jumps
  - Conditional jumps
    - *Which branch should enter the pipeline?*
  - Longer pipe → bigger problem
    - Decision in the middle/end of pipe
    - Pipe emptying up to few tens of clk’s

μp acceleration

- Branch prediction
  - Possible solutions
    - Branch prediction
      - $P_{success}<1$
    - Multipath execution
      - Hardware multiplication
      - Implicit execution (no result publication) until it’s known which path is correct
      - Good example: Intel Itanium

µp acceleration

- Branch prediction
  - Branch target buffer



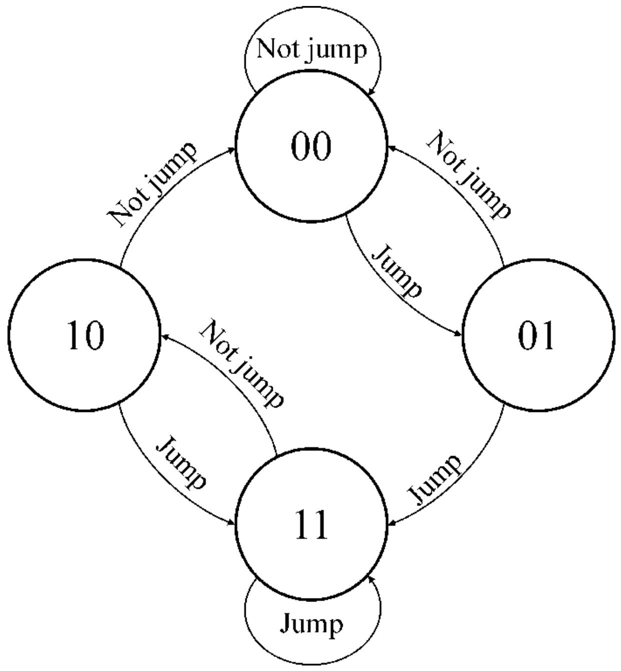
BTB size	Hit ratio
16	40%
32	50%
64	64%
128	72%
256	78%
512	80%
1024	85%
2048	87%

µp acceleration

- Branch prediction
  - Branch prediction methods
    - Static
      - Command bit defined by a compiler
        - » Based on possible code execution analysis
        - » *What if a compiler makes a mistake?*
      - By jump address (Intel’s solution: Pentium III)
        - » Negative (jump back): end of loop → jump
        - » Positive (jump forward): error service → not jump

µp acceleration

- Branch prediction
  - Branch prediction methods
    - Dynamic
      - History bits set according to program flow
        - » 1 bit: too little („checkerboard”)
        - » 2 bits:

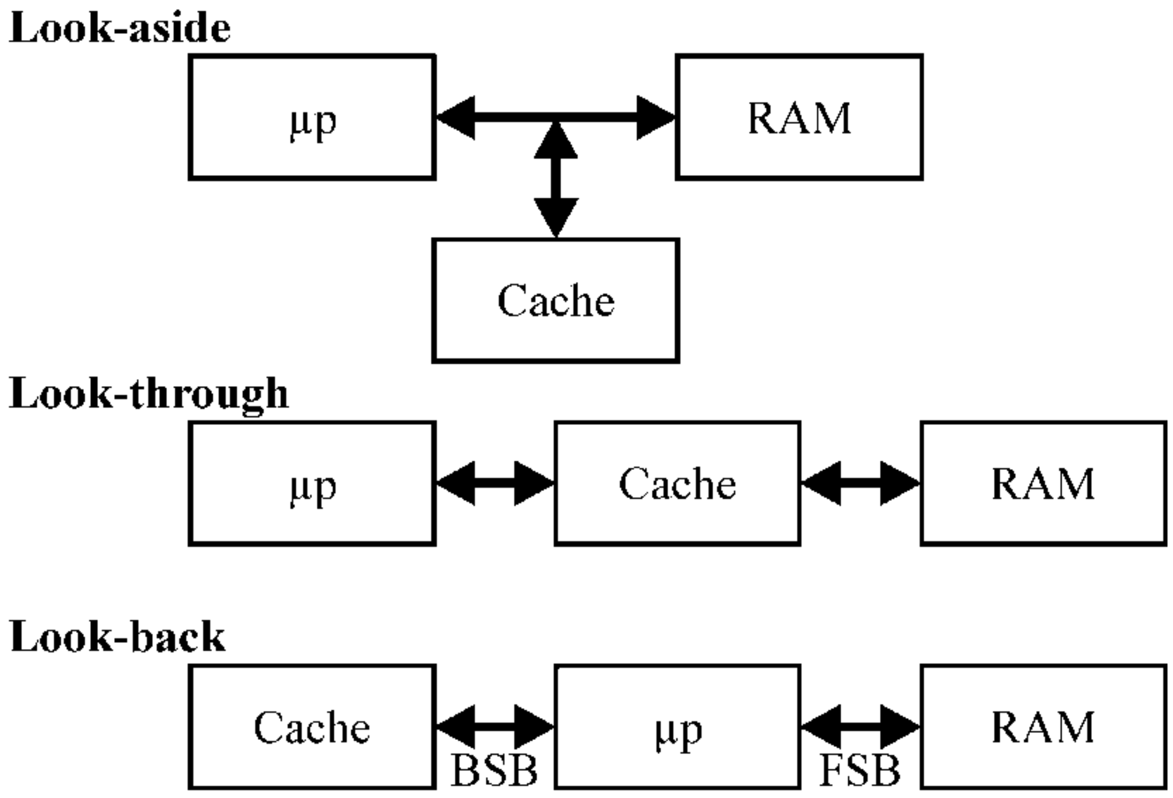


µp acceleration

- Optimisation
  - For a given µp type
  - Synchronous pipes:
    - Manual instr. placement for better pairing

µp acceleration

- Cache memory
  - Placement in a µp system

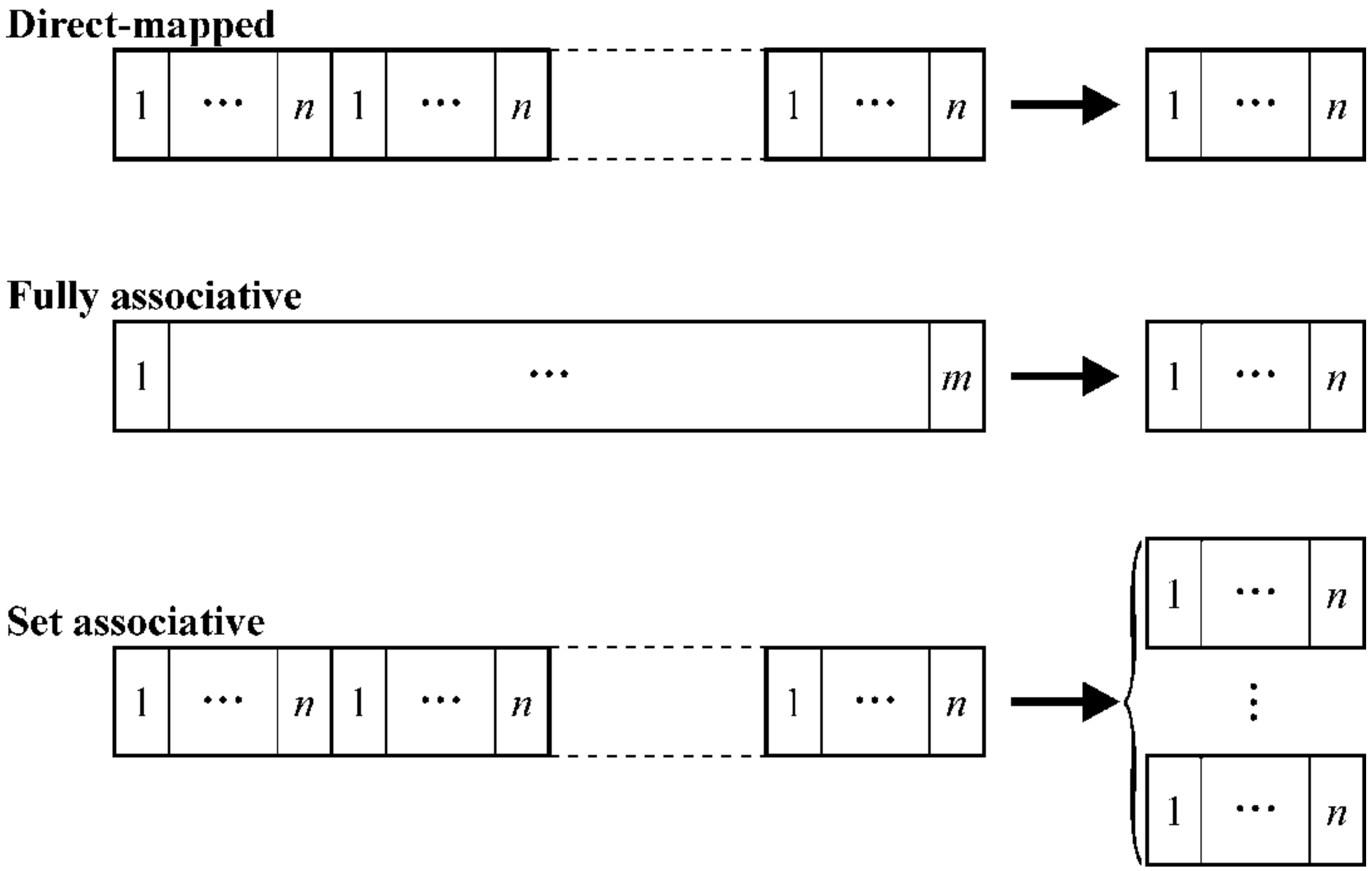


µp acceleration

- Cache memory
  - *What is better?*
    - Small capacity, high speed
      - Compact code
    - Large capacity, low speed
      - Distributed code

µp acceleration

- Cache memory
  - Cache organisation



μp acceleration

- Cache memory
  - Policies
    - Write-through
    - Write-back
  - *How to ensure cache & RAM consistency in a multiprocessor system?*
  - MESI protocol
    - Modified
    - Exclusive
    - Shared
    - Invalid