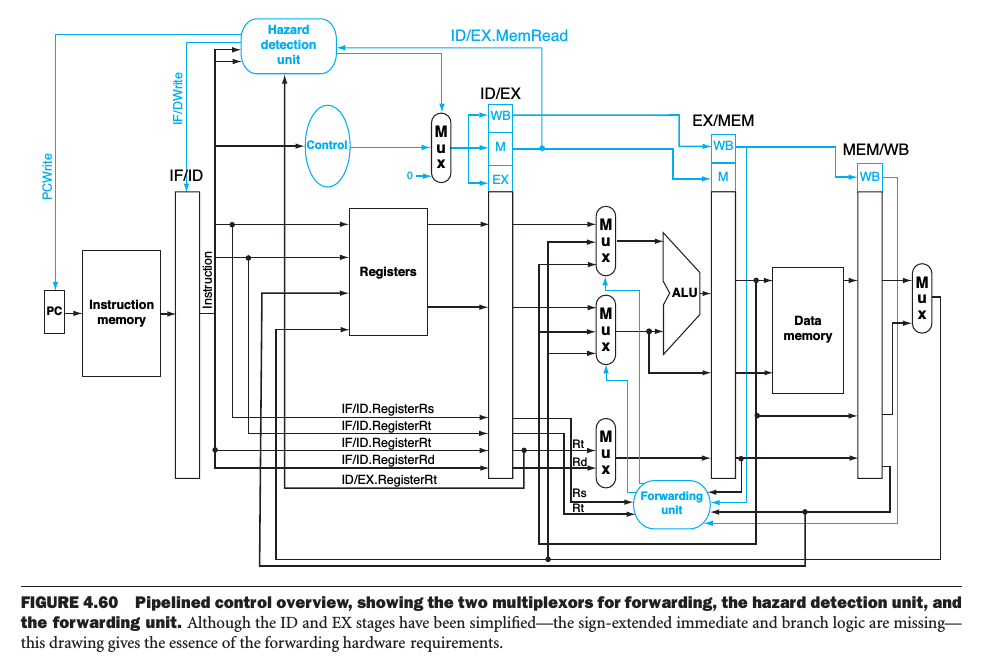
1. An overview of your processor optimization, including figures (e.g., block diagram, cache or branch predictor structure).
2. A description of your design approach, including testing and evaluation procedure.
3. Detail any optimizations you implemented, evaluation methodology, and results.
4. Conclusions and, if appropriate, post mortem detailing things you would do differently given the opportunity.

the approach taken to evaluate the optimization

1. Introduce the design problem.
2. Introduce your solution, in general terms, with a block diagram (if applicable).
3. Describe in some detail the different components of your system, including why you implemented particular elements.
4. Describe how you chose particular parameters and how you evaluated the system.
5. Describe your results and explain the behavior you observe.
6. Make recommendations based on your experience; would you do anything different if you were able to do the project over?

*Overview*

The processor is a standard five-stage 32-bit pipelined MIPS processor with forwarding unit and hazard detection unit. The optimization is based on the cache implementation, which includes instruction cache and data cache.



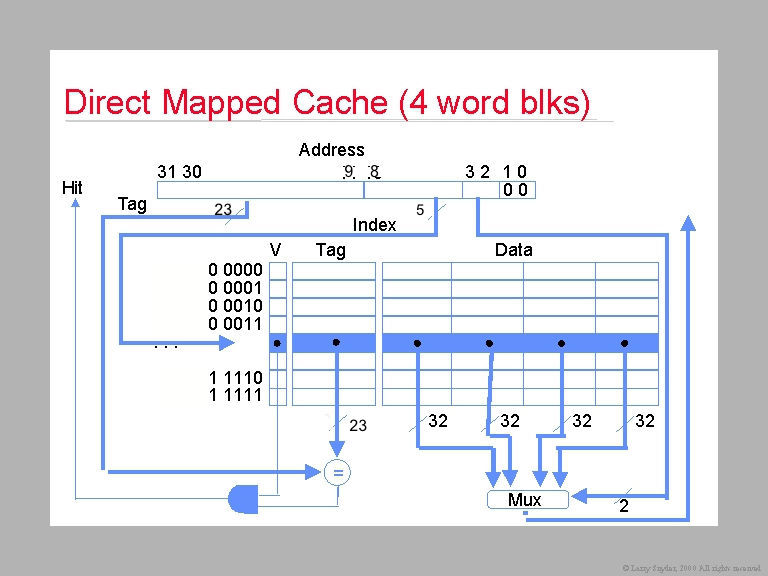
*Description*

*design approach*

For instruction cache, there is only read case. We separate working flow into three stages: idle and compute stage, memory access stage, and write-back and memory read stage. In the first stage, if hit exists, instruction cache directly outputs result. Otherwise, it will determine whether needs to conduct write-back based on dirty bit. The second stage is more like a control stage, in which several intermediate signals are set and reset. The third stage is actually for interacting

with memory, including write-back (if dirty) and reading from memory. Both requires four times to complete, that is, one word at a time. After write-back is conducted, it will go back to the first stage and follow the hit pattern.

For data cache, there are both read and write cases. However, the only change is in the first stage. If hit happens, data will be directly written into the block. Other steps are the same to instruction cache.



*testing procedure*

We applied several MIPS program to test the performance of processor.

Factorial program: it is used to calculate the factorial of an integer. The result is stored into register 2. The program contains basic ALU execution: add and multiply and uses branch instruction to conduct looping.

Bitwise program:

Fibonacci program: it is used to generate Fibonacci series. The generated numbers are first stored in register 2, and then to adjacent locations in memory. The program generates in total 32 numbers. Store word instructions show the advantage of cache for interacting with memory.

*evaluation procedure*

-compare the time consumed to run program

*Details*

*components description*

The pipeline basically divides a typical instruction into five stages, IF (*Instruction Fetch*), ID (*Instruction Decode*), EX (*Execution*), MEM(*Memory Access*), and WB (*Write Back*). Forwarding unit is a hardware solution to deal with data hazards, which is to pass proper values early from the pipeline registers to the input of ALU rather than waiting for WB to write register file.

Hazard detection unit is used in the ID stage to insert a stall between load and its use.

*optimization*

In the former project, we developed two memory: instruction memory and data memory, and put them into two stages: IF and MEM, and also assumed that it has no delay. In this project, memory is implemented as a unified one for both code and data, and it is isolated from stages as a separate entity. The memory delay is set as 10 cycles.

In the present work there are two types of caches included. **Instruction cache** is to speed up executable instruction fetch. **Data cache** is to speed up data fetch and store.

The cache implemented in our project has the following characteristics and parameters:

Write-back policy

Direct-mapped

32-bit words

128-bit blocks (4 words per block) word addressable

4096 bits of data storage (32 blocks)

32-bit addresses

data tags flags (valid bit, dirty bit)

Memory cache is the fastest type of memory after the CPU registers, therefore, the cache design and caching strategies directly impact processor performance.

The write method used is write-back policy. Since the cache storage is large(4KB), there are not many write-backs and the method is clearly better than the write-through method.

The structure used is direct-mapped. When the data required is in cache(hit), the efficiency of direct-mapped is fully exploited. Because it needs less time to search for data in cache compared to higher associativity.

*evaluation methodology*

*result*

*Conclusion*