CENG 3420 Quiz 2

Solutions

Q1 **[30 points]** Pipelined MIPS processor contains 5 stages: IF, ID, EX, MEM, and WB. Pipelline hazards consist of structural hazards, data hazards, and control hazards. The following code is run on a 5-stage MIPS pipeline with full forwarding.

```
lw   $t0, 0($a0)
add  $t0, $t0, 1
sw   $t0, 0($a0)
add  $a0, $a0, 4
```

1. List all the data hazards.
2. Which data hazards can NOT be resolved with forwarding?
3. Rewrite the code to eliminate stalls on the 5-stage pipeline with full forwarding.

1. `lw-add`: read-after-write (RAW) on `$t0`;

2. `lw-add`: this hazard is load memory then use, thus it cannot be resolved through forwarding.

3.
```
lw   $t0, 0($a0)
add  $a0, $a0, 4
add  $t0, $t0, 1
sw   $t0, -4($a0)
```

Q2 **[20 points]** A memory hierarchy includes a TLB and a cache. A memory reference can encounter three different types of misses: TLB miss, page table miss, and cache miss. The following table lists all combinations of three types of misses. Please complete the last colum on whether each combination can actually occur.

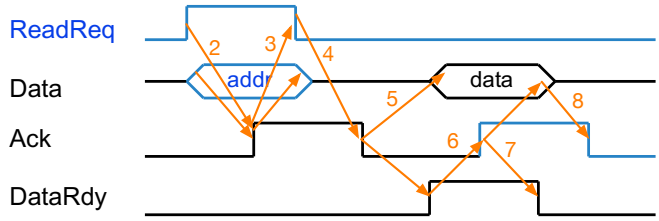
TLB	Page Table	Cache	Possible? Under what circumstances?
Hit	Hit	Hit	Yes
Hit	Hit	Miss	Yes
Hit	Miss	Hit	No
Hit	Miss	Miss	No
Miss	Hit	Hit	Yes
Miss	Hit	Miss	Yes
Miss	Miss	Miss	Yes
Miss	Miss	Hit	No

Q3 [30 points] In a direct mapped cache with **16KB** of data and **4**-word blocks, assume an address is 32-bits. (Note: KB = kilobyte)

1. In an address, what's the size of `index` field?
2. In an address, what's the size of `tag` field?
3. In a cache block there are 4 words. What's the size of one cache block?

1. 16KB = 4K words = 2^{12} words. Since each block contains 4 words, `index` field length $= 12-2 = 10$.
2. `tag` field length: $32-(10+2+2) = 18$.
3. The size of one cache block includes: 4 words ($4 \times 32 = 128$ bits), one `tag` field (18 bits), and one valid field (1 bit). Thus the total size of one cache block is $128+18+1 = 147$.

Q4 [20 points] Considering a scenario that data is transferred from memory to I/O devices. Complete the following Asynchronous Bus Handshaking Protocol.



1. I/O device requests by raising `ReadReq` & putting addr on the data lines
2. Memory sees `ReadReq`, reads addr from data lines, and raises `Ack`
3. I/O device sees `Ack` and releases the `ReadReq` and data lines
4. Memory sees `ReadReq` go low and drops `Ack`
5. When memory ready, putting data on data lines & raises `DataRdy`
6. I/O device sees `DataRdy`, reads data from data lines & raises `Ack`
7. Memory sees `Ack`, releases data lines, and drops `DataRdy`
8. I/O device sees `DataRdy` go low and drops `Ack`