CS61C Fall 2012 – 12 – ECC & Interrupts

Hamming Codes

Recall the basic structure of a Hamming code. Given bits 1...,m, the bit at position 2^n (starting at n=0, the first bit) is parity for all the bits with a 1 in position n. For example, the first bit is chosen such that the sum of all odd-numbered bits is even.

Q1. Suppose you had the bits 0011 and we want to add some bits to allow single error correction.

a) How many bits do we have to add?

3

b) Which bits are parity bits?

Marked with underscore -> _ _ 0 _ 0 1 1

c) Which bits does each parity bit cover?

bit 1: 1, 3, 5, 7

bit 2: 2, 3, 6, 7

bit 3: 4, 5, 6, 7

d) Write the completed coded representation for this bit string.

1000011

e) What do we need to do to make this into a SEC-DED code?

Add another parity bit over the whole sequence.

Q2. Find the original bits given the following SEC Hamming Code.

a) 0110111

Group 1 error, group 2 ok, group 4 error -> bit 5 erroneous; should be 0.

Answer: 1011.

b) 1001000

Group 1 error, group 2 ok, group 4 error -> bit 5 erroneous; should be 1.

Answer: 0100.

Q3. If we didn't need SEC but wanted SED, how many bits would we need to add for 4 bits?

1 (for parity check).

Interrupt

Definition: "An unscheduled event that disrupts program execution"

- Hardware (asynchronous): I/O service, timer expiration, hardware failure, etc.
- Software (synchronous): Exceptions/ traps.
  Examples: divide-by-zero, overflow, invalid memory address, page fault.

Handling an interrupt:

- Flush pipeline in case of exception (bubble out the following instructions)
- Disable interrupts (setting interrupt enable bit to 0)
- Save PC of the offending instruction in "Exception Program Counter" (EPC)
- Jump to interrupt service routine/ interrupt handler
- After exception/ interrupt is handled, jump to EPC and resume execution

Q4. What would be an alternative to using interrupts for interacting with a (much slower) device? Keeping this in mind, why is an interrupt-based approach efficient?

Busy waiting/polling: the processor is responsible for continually/periodically checking if any I/O device has completed any request. If we use interrupts, the processor is free to do other work independently and has greater freedom in scheduling the service of device interrupts.

Q5. For each of the 5 MIPS pipeline stages, list exceptions that could occur.

Pipeline Stage	Problem exceptions occurring
IF	Page fault on instruction fetch, misaligned memory access, memory protection violation
ID	undefined or illegal opcode
EX	arithmetic exception
MEM	page fault on data fetch, misaligned memory access, memory protection violation
WB	None

Q6. An *inter-processor interrupt* (IPI) is a type of interrupt where one processor will interrupt another. What are some situations where this could be used?

- Synchronize the cache and memory management hardware
- Updating TLB
- System crash/shutdown

Q7. An *interrupt storm* is when a processor is bogged down by a huge number of rapid interrupts (usually due to a faulty device or configuration). How could one mitigate this problem?

"Rate limiting": Device controllers must wait a certain amount of time before generating another interrupt... obviously this amount of time requires careful tuning...