Announcements

- Program part A was due at 10:30.
- Program part B was handed out.
- Reading for this lecture:
  - Mano sections 8-7, 12-5 to 12-7

Program Control Instructions

- How do we create instructions that branch based on the relative values of two registers?
  - Answer: add status bits
- Status Bits
  - reflect the state of the last arithmetic operation
  - common status (condition) bits
    - Zero - result all zeros
    - Carry - carry out of the last bit position
    - Sign - is the sign bit set in the result?
    - V - overflow (xor of carry in and carry out of sign bit)
  - compare instruction
    - computes A - B
    - does not store result
    - updates status bits

Using Status bits for Branching (Signed comparison)

Instruction	Status Bits
BGT	S’ ^ Z’
BGE	S’
BLT	S
BLE	S v Z
BE	Z
BNE	Z’

Computer Architecture

- What is a processor architecture?
  - Organization of a processor as seen by software
    - number of size of registers
    - instructions
      - what instructions are built into hardware
    - addressing modes
      - indirection
      - auto increment modes
    - size of memory
  - It is different than the implementation
    - internal registers
    - timing of instructions
    - actual amount of memory installed/available

Instruction Formats

The format of the instruction tells us a lot about an architecture

- OpCodes
  - indicates the types of operations the machine can perform
  - complexity of instructions
    - can they operate on one ore multiple words of memory?
    - simpler instructions are easier to make run fast
- Instruction length
  - dictates available bits for op-codes and operands
  - can be fixed (MIPS, Mano) or variable length (Intel)
    - fixed length instructions are easier generally faster
- Number of Registers
  - many registers (MIPS)
  - one register (Mano)
  - zero registers (stack machines)

RISC Architecture

- Reduced Instruction Set Computer
- Sometimes less is more!
- Put fewer features in hardware, but make them run very fast
  - only include common instructions in hardware
  - eliminate most addressing modes
    - use load/store instead
- Use lots of registers
- Fixed length instructions
- Most instructions require one cycle to execute
- Computer Architecture based on quantitative studies
  - measure real programs and find out what is important

Registers vs. Memory

- Registers are very fast
  - typically registers can be used in one clock pulse while memory might take 20 cycles
- Memory is large
  - at most a few hundreds registers in a machine, but memory can be megabytes to gigabytes
- Consider the number of memory references that are required for different styles of architecture

Multi-Register (RISC)

```
*   load R1, A
*   load R2, B
*   load R3, C
*   load R4, D
    add R1, R1, R2
    add R3, R3, R2
    mul R1, R1, R3
*  store X, R1
```

5 memory references

One Address

```
*   load A
*   add B
*   store T
*   load C
*   add D
*   mult T
*   store X
```

7 memory references

Stack (zero address)

```
2   push A  (rd a, wr stack)
2   push B (rd b, wr stack)
3   add    (pop,pop,push)
2   push C
2   push D
3   add
3   mul
2   pop x
```

19 memory references

Time to Execute Sample Program

- RISC
  - 8 * 21 + 20 * 5 = 268 cycles
- One Address
  - 7 * 21 + 20 * 7 = 287 cycles (7% slower)
- Stack
  - 8 * 21 + 20 * 19 = 548 cycles (104% slower)

Computer Memory

- Experience shows memory is not used evenly
- Locality of references
  - spatial locality: If use memory location x, then it is likely you will use memory near location x.
    - programs tend to execute several instructions in a row
    - array and structure references
  - temporal locality: if memory location x is used at time t, it is likely that it will be used again shortly within time t + epsilon
    - loops in program
    - updates to the same variable
- Goal: Make access to commonly used memory locations faster

Cache Memory

- A fast place to keep frequently used information
  - sort of like a personal telephone book vs. city book
  - caches have a fixed size that is much smaller than main memory
- If look for a value and its in the cache, it's a hit
- If look for a value and its not in the cache, it's a miss
- Key criteria for a cache is the hit ratio
  - hit ratio = $\dfrac{\text{\# of hits}}{\text{\# of hits + \# of misses}}$
  - possible to get a hit ratio of over 90%
  - if a hit takes 10ns and a miss 100ns, then a hit rate of 90% means an effective memory access time of 20 ns (or 5 times faster than without the cache!)

Questions about Cache Memory

- How big is the cache?
- How many caches are there?
  - first level caches are typically 8 to 64k (often on the processor chip)
    - due to propagation delays
      - in 1 ns electricity only travels about a foot
      - current clock speed on available machines is < 3ns
  - second level caches can be over a megabyte
- What to do if there is no free space in the cache?
  - need to remove something (called replacement or eviction)
  - FIFO, LRU, and random are all used
- How do we find something in the cache?
  - needs to be fast!
  - several ways to find data in a cache