Memory

- `pc` - program counter
  - starts at 0x0040 0000
  - keeps track of the address of the next instruction to execute
    - goes up in increments of 4
    - When you use a jump instruction, the program counter jumps to that address
- `$gp` - global counter
  - starts at 0x1000 8000
- `$sp` - stack pointer
  - starts at highest address
    - 0x7fff fffc

Memory Hierarchy

- Principles of locality applied as memory hierarchy

Temporal Locality

- If a data location is referenced then it will tend to be referenced again soon

Spatial Locality

- If a data location is referenced, then data locations with nearby addresses will tend to be referenced soon

Data

- Hierarchical
  - A level closer to the processor is generally a subset of any level further away, and all the data is stored at the lowest level
  - Data is copied between only upper and lower levels at any given time
  - The min unit of info that can be either present or not present in the two-level hierarchy is called a **block** or **line**

Performance

Hit Rate

- Fraction of memory accesses found in a level

Miss Rate

- (1 - hit rate)
- Fraction of memory accesses not found in a level

Hit Time

- Time required to access a level of the memory hierarchy + time to determine whether the access is a hit or miss

Miss Penalty

- Time required to fetch a block into a level of the memory hierarchy from the lower level + time to access the block, transmit it between levels, insert it in the level that experienced the miss, and then pass the block to the requestor

MIPS Perspective

- First part, near bottom of address space (starting at 0x0040 0000) is the text segment, which holds the program's instructions
- Second part is the data segment divided into two parts
  - Static data (starting at 0x1000 0000) contains objects whose size is known to the compiler and whose lifetime (interval during which a program can access them) is the program's entire execution
  - Dynamic data - allocated by program as it executes
    - e.g. C `malloc` finds and returns a new block of memory
- Third part is program stack segment
  - Starts at 0x7fff fffc
  - Like dynamic data, the max program stack size is not known in advance
  - As the program pushes values on to the stack, the OS expands the stack segment down toward the data segment
    - e.g. call procedures

Stack Operations

- Stack is word accessible
- Stack is operated via push and pop
  - Neither push or pop are native operations
- You can use stack pointer instead

Push Operation

- Store data into address pointed by $sp

Pop Operations

Load and Store Instructions

Load

- Load word
- Loads content in specified memory address
- `lw d, off(b)`
    - You can specify a register with an address as base

One Machine Cycle Delay

- There is a delay, so you need a no op after the lw instead of directly using the register

Store

- Store word
- Stores content from a register into main memory
- `sw t, off(b)`

Alignment Restrictions

- When a word (4 bytes) is loaded from or stored into the memory, the address must be a multiple of 4
- This restriction makes the hardware simpler and faster

Sign Extension

Sign Extended

- This is what load word does

Zero Extended

Endian

Little Endian

- Little end first
    - Smallest place value first
- Example
    - if `$gp` is pointing to 0x0040 0000
    - Going upwards in memory
        - EF
        - BE

        - AD
        - DE

Big Endian

- Big end first
    - Biggest place value first
- Example
    - if `$gp` is pointing to 0x0040 0000
    - Going upwards in memory
        - DE
        - AD
        - BE
        - EF

Loading a single byte

lb t, off (b)

- $t <- sign-extended byte
    - stuff leading side with f if leading binary bit is 1, 0 otherwise
    - ex. `lb $t1 0x4($t0)`
        - `0xc00lface` is in value +4 (so 7, 6, 5 , 4)
        - if this is `ce`
        - register `$t1` will be `0xffffffce` due to leading bit (left side) being 1
- b is base register
- off is 16-bit two's complement

lbu t, off (b)

- $t <- zero-extended byte
    - With above example, we would get `0x000000ce` due to sign extension
    - always filled with 0
- from memory address b + off
- b is a base register
- off is 16-bit two's complement

Loading a half word

lh t, off (b)

- sign extended

- starting at memory b offset
- offset is 16 bit twos complement

lhu t, off (b)

- zero-extended