15 MIPS

Friday, October 9, 2015 10:03 AM

MIPS

- 1985, 10 years after 6502
- 32 bit CPU
  - 32 bit Address and Data
  - 4GB memory
  - 32 32-bit registers
    - Only 2 that have fixed purpose in the hardware
      - $0 or $zero
        - Always 0, always has the value 0
        - If you write to, it just disappears because whenever you read from it, it reads as 0
      - $31 or $ra
        - Return address register
    - The rest are deemed by convention, but you can do w.e. you want with it
      - $1 or $at
        - Assembler temporary
      - $2, $3 or $v0, $v1
        - Result of a function called result registers
        - If function returns 32bit answers, goes to $2, if 64, goes to both
      - $4 - $7, $a0-$a3
        - Argument registers
        - Parameters for functions
        - (good functions have less parameters, functions should do one thing) - Peter
      - $8 - $15, $t0 - $t7
        - temp
        - 8 registers for you to do w.e. heck you want to do
        - If a function, can use the t register, not responsible for saving it. Coder should have saved it
      - $16 - $23, $s0 - $s7
        - S registers, temps
        - Saved temporary registers
        - Caller will keep this register
      - $24, $25, $t8, $t9
        - temp
      - $26, $27
        - Kernel registers
        - Also called $k0, $k1
        - officially there for the operating system to use, but can't rely on them, OS may trash
      - $28, $29, $30 functions
  - Instructions

caller | callee
$tx | $sx

- Add
    - $t7 destination
    - $t1, $t1 sources
- **R format**
    - All instructions are 32-bits long
        - Makes decoding very straight forward
    - 31-26 opcode
    - 5 bit each
        - 25-21 source
        - 20-16 source
        - 15-11 destination
    - Shift amount
    - Function field
- "Addi $t7, $t2, 25"
    - Putting a literal number
- **I Format**
    - 31-26 Opcode
    - 25-21 source
    - 20-16 destination
    - 15-0 immediate, basically the number 25
        - But only 16-bits! Can only add 16 bit numbers
- I loop
    - Jumping to a label called loop
    - 31-26 opcode
    - 25 - 0 address
        - Problem: there are 32-bit addresses that exist
        - Trick, address has to be divisble by 4
            - If divisible by four, bottom two bits always 0
            - So just leave them out