T-Mobile’s Wi-Fi / Cell phone ⇒ T-mobile just announced a new phone that can operate normally as a cell phone, but when it’s near a “T-mobile wireless hotspot” (you could own at home), it switches to Wi-Fi & calls are FREE. Yes, FREE.

Five Components of a Computer

Datapath Summary
- The datapath based on data transfers required to perform instructions
- A controller causes the right transfers to happen

Controller

CPU clocking
For each instruction, how do we control the flow of information through the datapath?
- Single Cycle CPU: All stages of an instruction are completed within one long clock cycle.
  - The clock cycle is made sufficient long to allow each instruction to complete all stages without interruption and within one cycle.

Outline of Today’s Lecture
- Design a processor: step-by-step
  - Requirements of the Instruction Set
  - Hardware components that match the instruction set requirements
  - Partial Datapath in detail

How to Design a Processor: step-by-step
1. Analyze instruction set architecture (ISA) ⇒ datapath requirements
   - meaning of each instruction is given by the register transfers
   - datapath must include storage element for ISA registers
   - datapath must support each register transfer
2. Select set of datapath components and establish clocking methodology
3. Assemble datapath meeting requirements
4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer.
The different fields are:
• **op**: operation ("opcode") of the instruction
• **rs, rt, rd**: the source and destination register specifiers
• **shamt**: shift amount
• **funt**: function selects the variant of the operation in the "op" field
• **address / immediate**: address offset or immediate value
• **target address**: target address of jump instruction

### Step 1a: The MIPS-lite Subset for today
- **ADDU and SUBU**
  - `addu rd, rs, rt`
  - `subu rd, rs, rt`

- **OR Immediate**
  - `ori rt, rs, imm16`

- **LOAD and STORE Word**
  - `lw rt, rs, imm16`
  - `sw rt, rs, imm16`

- **BRANCH**
  - `beq rs, rt, imm16`

### Step 1: Requirements of the Instruction Set
- **Memory (MEM)**
  - instructions & data (will use one for each)
- **Registers (R): 32 x 32**
  - read RS
  - read RT
  - Write RT or RD
- **PC**
  - Extender (sign/zero extend)
  - Add/Sub/OR unit for operation on register(s) or extended immediate
  - Add 4 or extended immediate to PC

**Compare registers?**

### Step 2: Components of the Datapath
- **Combinational Elements**
- **Storage Elements**
  - Clocking methodology

### Register Transfer Language
**RTL** gives the meaning of the instructions

```
{op, rs, rt, rd, shamt, funct} <---- MEM[ PC ]
{op, rs, rt, Imm16} <---- MEM[ PC ]
```

```plaintext
All start by fetching the instruction
```

**inst**
```
ADDU  R[rd] ← R[rs] + R[rt];  PC ← PC + 4
SUBU  R[rd] ← R[rs] − R[rt];  PC ← PC + 4
ORI   R[rt] ← R[rs] · zero_ext(imm16);  PC ← PC + 4
LOAD  R[rt] ← MEM[ R[rs] + sign_ext(imm16) ];  PC ← PC + 4
STORE MEM[ R[rs] + sign_ext(imm16) ] ← R[rt];  PC ← PC + 4
BEQ   if ( R[rs] == R[rt] ) then
else PC ← PC + 4
```

### Combinational Logic Elements (Building Blocks)
- **Adder**
- **MUX**
- **ALU**
ALU Needs for MIPS-lite + Rest of MIPS

- Addition, subtraction, logical OR, ==:
  - ADDU R[rd] = R[rs] + R[rt]; ...
  - SUBU R[rd] = R[rs] - R[rt]; ...
  - ORI R[rt] = R[rs] | zero_ext(Imm16)...
  - BEQ if (R[rs] == R[rt])...

- Test to see if output == 0 for any ALU operation gives == test. How?
- P&H also adds AND, Set Less Than (1 if A < B, 0 otherwise)
- ALU follows chap 5

Administrivia

- Read the book! Important to understand lecture and for project.
- P&H 5.1-5.4

Storage Element: Idealized Memory

- Memory (idealized)
  - One input bus: Data In
  - One output bus: Data Out
  - Memory word is selected by:
    - Address selects the word to put on Data Out
    - Write Enable = 1: address selects the memory word to be written via the Data In bus
  - Clock input (CLK)
    - The CLK input is a factor ONLY during write operation
    - During read operation, behaves as a combinational logic block:
      - Address valid $\Rightarrow$ Data Out valid after “access time.”

Storage Element: Register (Building Block)

- Similar to D Flip Flop except
  - N-bit input and output
  - Write Enable input
  - Write Enable:
    - negated (or deasserted) (0): Data Out will not change
    - asserted (1): Data Out will become Data In on positive edge of clock

Storage Element: Register File

- Register File consists of 32 registers:
  - Two 32-bit output busses: busA and busB
  - One 32-bit input bus: busW
  - Register is selected by:
    - RA (number) selects the register to put on busA (data)
    - RB (number) selects the register to put on busB (data)
    - RW (number) selects the register to be written via busW (data) when Write Enable is 1
  - Clock input (clk)
    - The clk input is a factor ONLY during write operation
    - During read operation, behaves as a combinational logic block:
      - RA or RB valid $\Rightarrow$ busA or busB valid after “access time.”

Step 3: Assemble DataPath meeting requirements

- Register Transfer Requirements $\Rightarrow$ Datapath Assembly
  - Instruction Fetch
  - Read Operands and Execute Operation
### 3a: Overview of the Instruction Fetch Unit

- The common RTL operations
  - Fetch the Instruction: \( \text{mem}[PC] \)
  - Update the program counter:
    - Sequential Code: \( PC \leftarrow PC + 4 \)
    - Branch and Jump: \( PC \leftarrow "\text{something else}" \)

![Diagram showing the instruction fetch unit](image)

- Instruction Word
  - Address
  - Instruction Word
  - Next Address Logic

### 3b: Add & Subtract

- \( R[rd] = R[rs] \oplus R[rt] \)
  - Ex.: \( addU \ rd, rs, rt \)
- \( Ra, Rb, \) and \( Rw \) come from instruction’s Rs, Rt, and Rd fields
- ALUctr and RegWr: control logic after decoding the instruction

![ALU and Register write logic](image)

### Peer Instruction

- Our **ALU** is a synchronous device
- We should use the main ALU to compute \( PC = PC + 4 \)
- The **ALU** is inactive for memory reads or writes.

### How to Design a Processor: step-by-step

1. Analyze instruction set architecture (ISA) ⇒ datapath requirements
   - meaning of each instruction is given by the register transfers
   - datapath must include storage element for ISA registers
   - datapath must support each register transfer
2. Select set of datapath components and establish clocking methodology
3. **Assemble** datapath meeting requirements
4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer.
5. Assemble the control logic (hard part!)