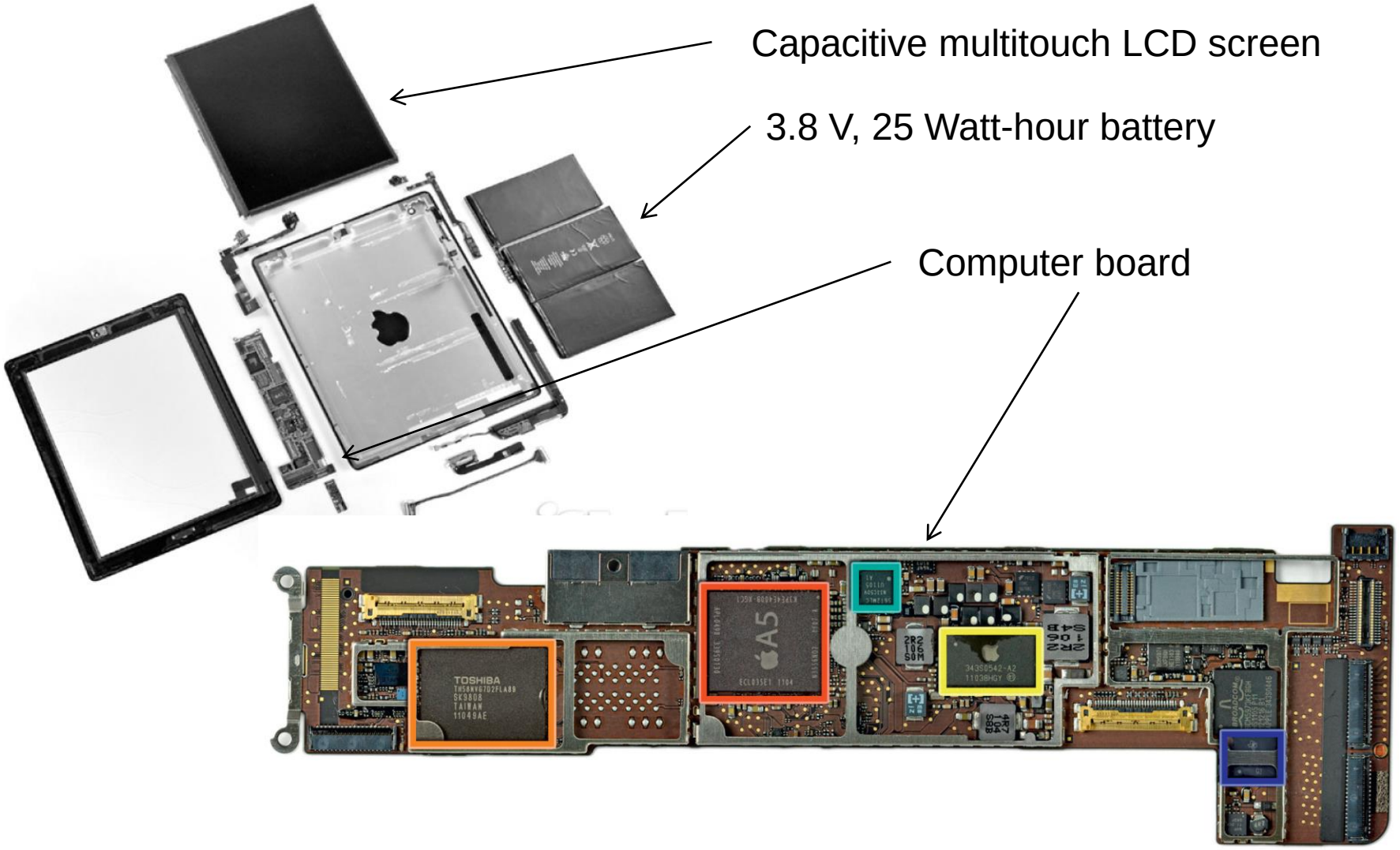
ECE/CS 472/572 Computer Architecture: Background

Prof. Lizhong Chen

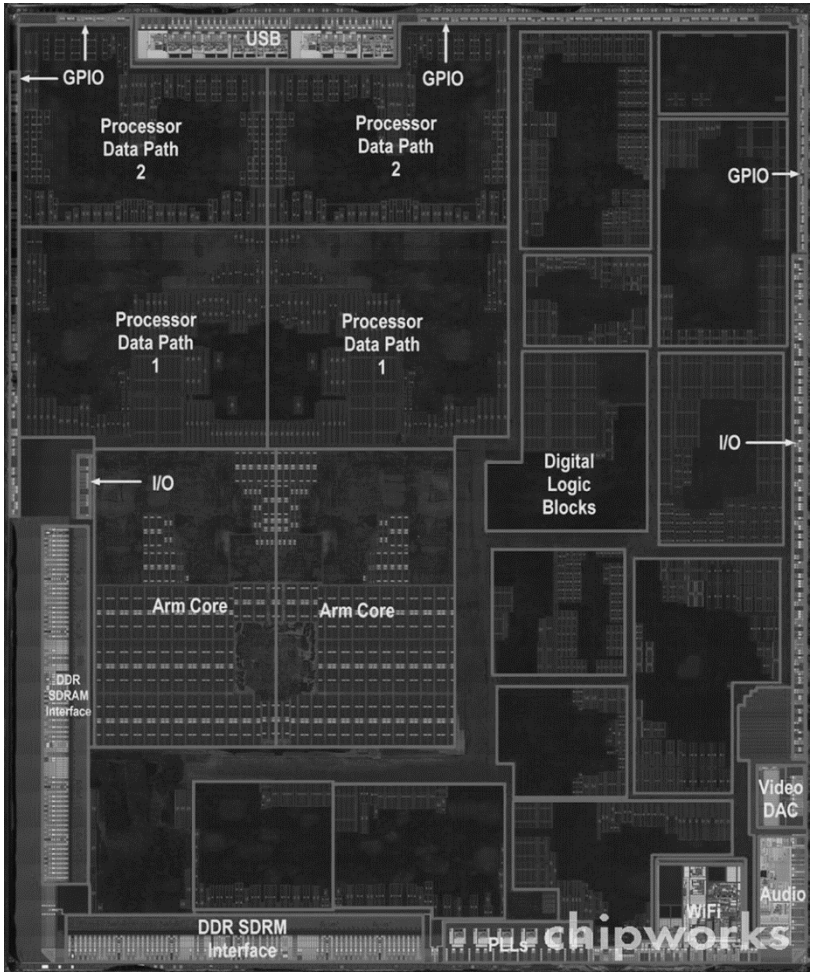
Spring 2019

Opening the Box



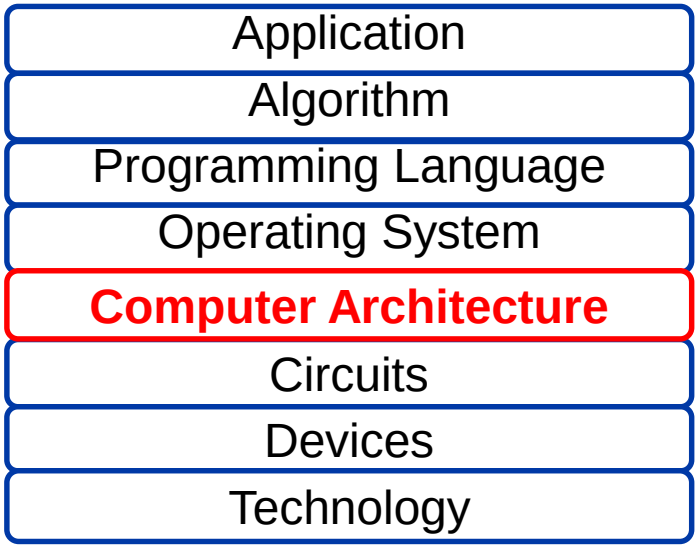
Inside the Processor

- Apple A5

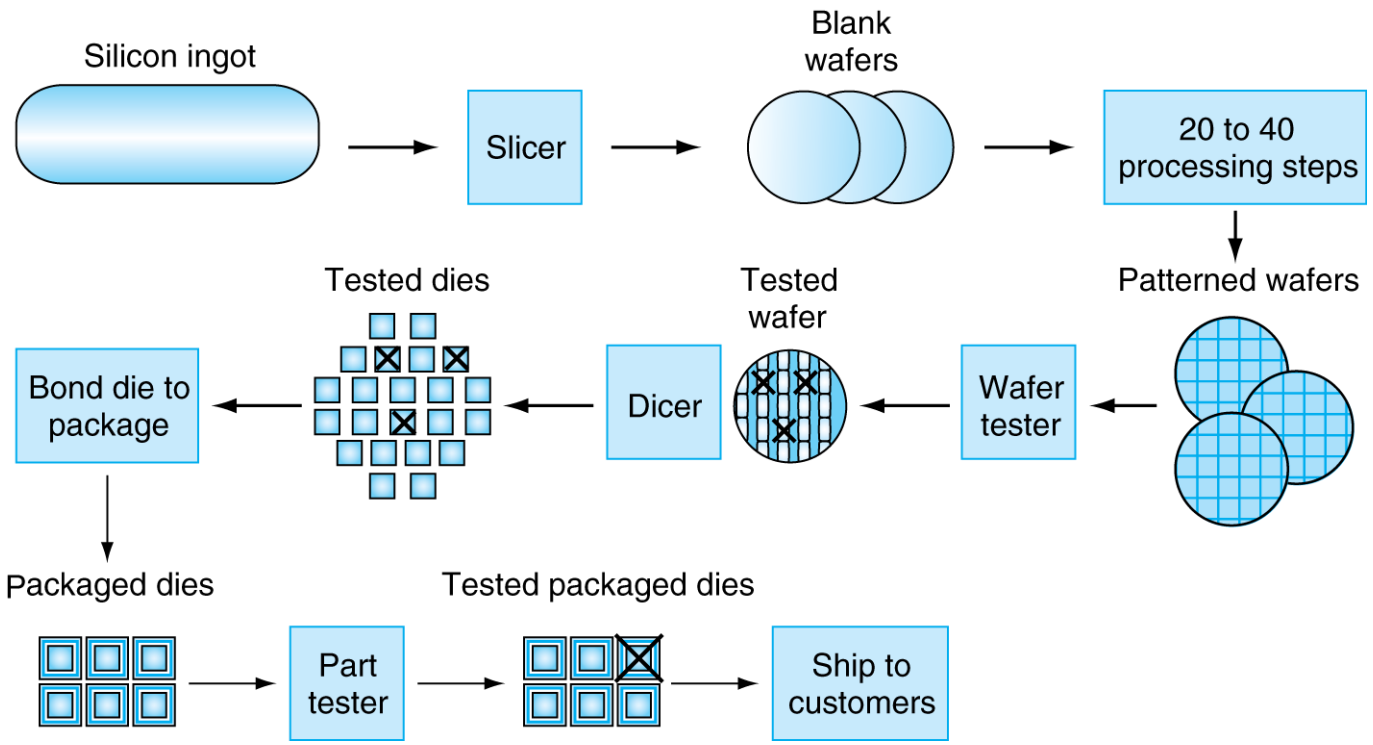


What is Computer Architecture ?

- **Computer Architecture** is the science and art of selecting and interconnecting hardware components to create computers that meet functional, performance and cost goals.



How Computer Is Made



https://www.youtube.com/watch?v=UvluuAIiA50

Relative Performance

- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time$_B$ / Execution Time$_A$
    = 15s / 10s = 1.5
  - So A has a speedup of 1.5 over B
- Define: Performance = 1/Execution Time
- "X is *n* time faster than Y"

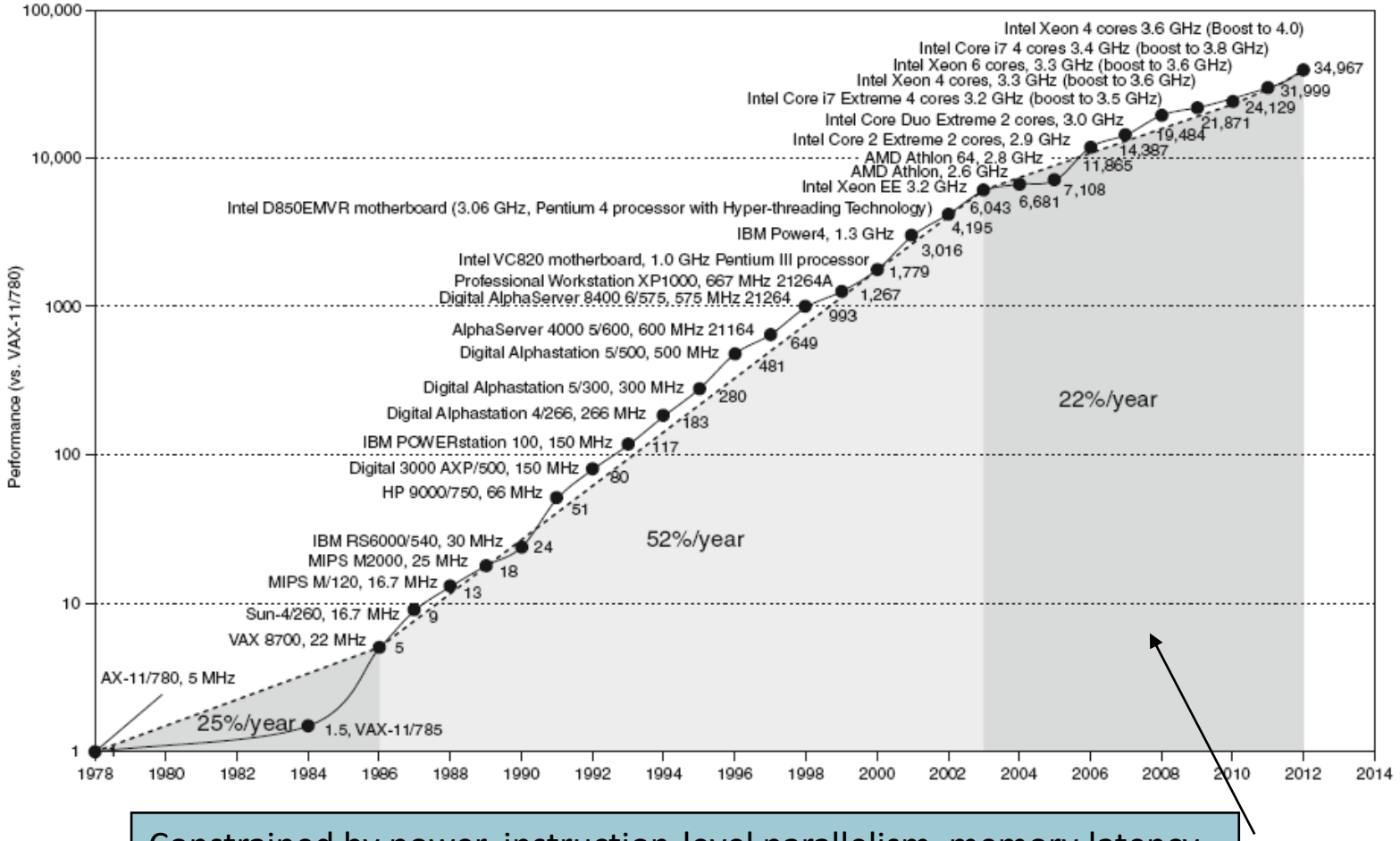
$$\text{Performance}_X / \text{Performance}_Y$$
$$= \text{Execution time}_Y / \text{Execution time}_X = n$$

Instruction Count and CPI

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

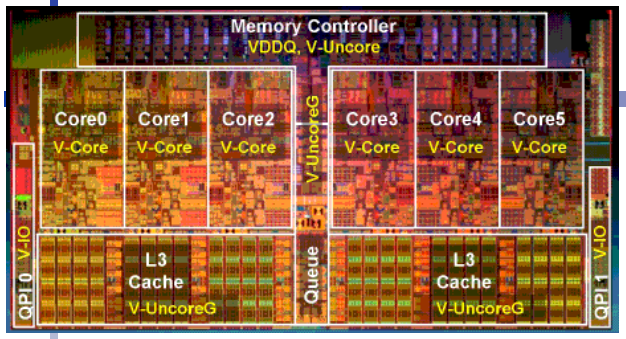
- Instruction Count for a program
  - Determined by program, ISA and compiler
- Average cycles per instruction (CPI)
  - Determined by CPU hardware
  - Different instructions may have different CPI
    - Average CPI affected by instruction mix

Uniprocessor Performance

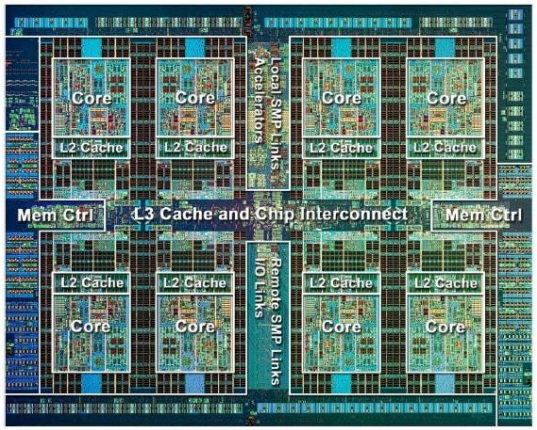


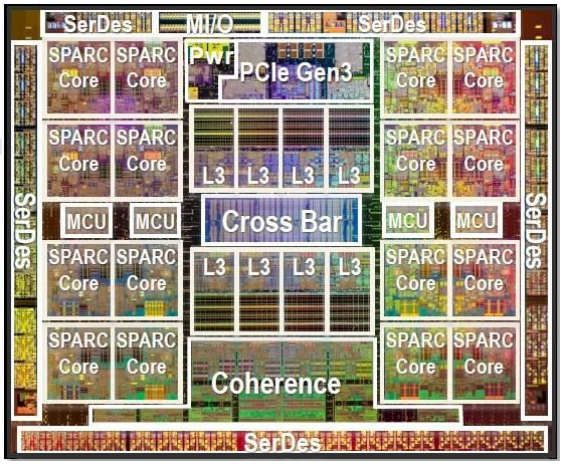
Constrained by power, instruction-level parallelism, memory latency

Intel Tick-Tock Model



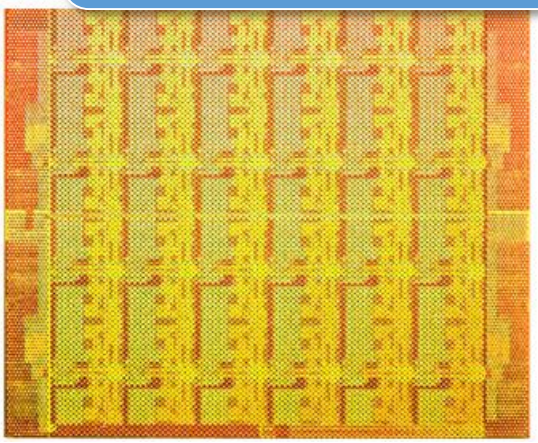
Intel Westmere-EP: 6-core[1]



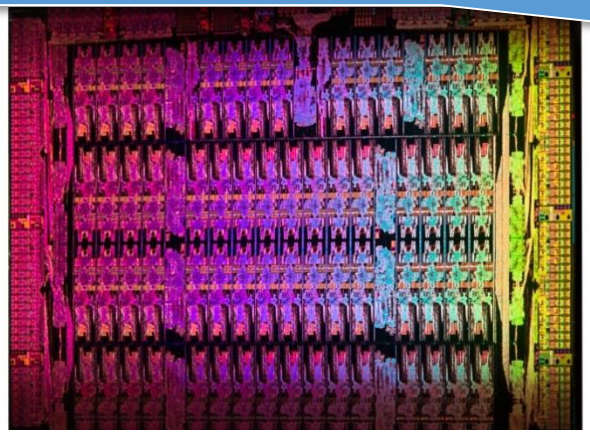


e[3]

Challenge: On-chip communication for parallel computing



Intel SCC: 48-core[4]



Intel Xeon Phi: 60-core[5]

ed processors

- Broadcom XLP-II: 20-core
- Cavium Octeon: 48-core
- Tilera Tile-Gx8072: 72-core

- Mobile devices – MPSoCs
  - CPU, GPU, DSP, etc.
- (GP)GPUs
  - Nvidia Kepler: 192x15 cores
  - AMD Liverpool: 1152 cores

[1] http://www.theregister.co.uk/2010/02/03/intel_westmere_ep_preview/
[2] http://www.theregister.co.uk/2012/10/03/ibm_power7_plus_server_launch/
[3] http://www.theregister.co.uk/2012/09/04/oracle_sparc_t5_processor/
[4] http://www.intel.com/pressroom/archive/releases/2009/20091202comp_sm.htm
[5] http://www.scientificcomputing.com/news/2013/02/intel-xeon-phi-coprocessor/

Instruction Set

- The repertoire of instructions of a computer
- Different computers have different instruction sets
  - But with many aspects in common
- Early computers had very simple instruction sets
  - Simplified implementation
- Many modern computers also have simple instruction sets
  - CISC vs. RISC

The MIPS Instruction Set

- Large share of embedded core market
  - Applications in consumer electronics, network/storage equipment, cameras, printers, …
  - Typical of many modern ISAs (see Appendixes E)
- We will examine two implementations of MIPS ISA
  - A simplified version
  - A more realistic pipelined version
- Simple subset, shows most aspects
  - Memory reference instructions: `lw`, `sw`
  - Arithmetic-logical instructions : `add`, `sub`, `and`, `or`, `slt`
  - Control transfer instructions: `beq`, `j`

MIPS Instruction Examples

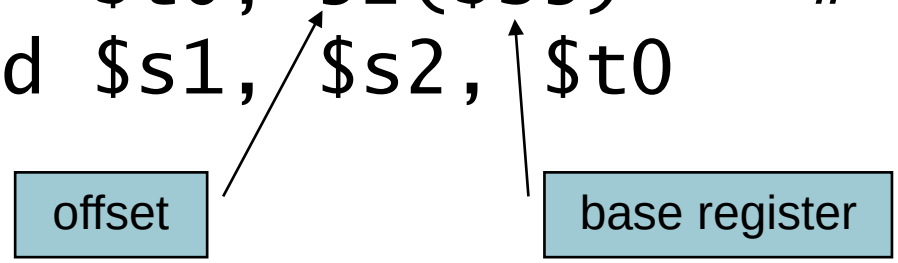
- C code:

```
g = h + A[8];
```

  - g in $s1, h in $s2, base address of A in $s3

- Compiled MIPS code:
  - Index 8 requires offset of 32
    - 4 bytes per word

```
lw  $t0, 32($s3)    # load word
add $s1, $s2, $t0
```



MIPS Instruction Examples

- C code:

```
A[12] = h + A[8];
```

  - h in $s2, base address of A in $s3

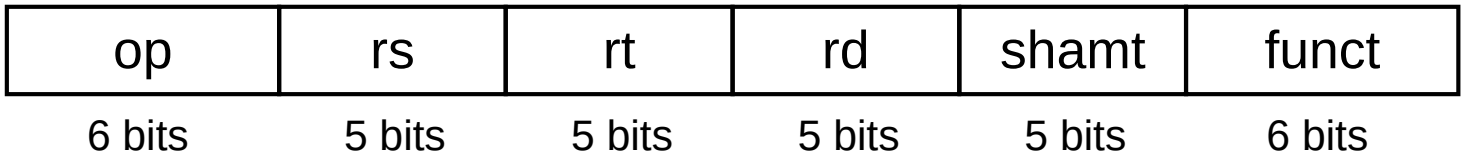
- Compiled MIPS code:
  - Index 8 requires offset of 32

```
lw  $t0, 32($s3)    # load word
add $t1, $s2, $t0
sw  $t1, 48($s3)    # store word
```

MIPS Instruction Examples

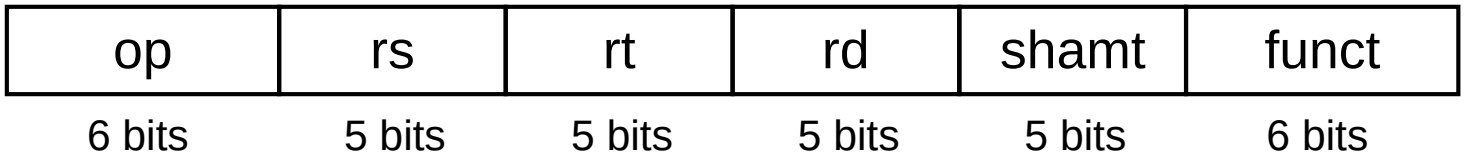
- Conditional
  - Branch to a labeled instruction if a condition is true; otherwise, continue sequentially
  - `beq rs, rt, L1`
  - If (rs == rt) branch to instruction labeled L1;
- Unconditional
  - `j L1`
  - Unconditional jump to instruction labeled L1

MIPS R-format Instructions



- Instruction fields
  - op: operation code (opcode)
  - rs: first source register number
  - rt: second source register number
  - rd: destination register number
  - shamt: shift amount (00000 for now)
  - funct: function code (extends opcode)

R-format Example

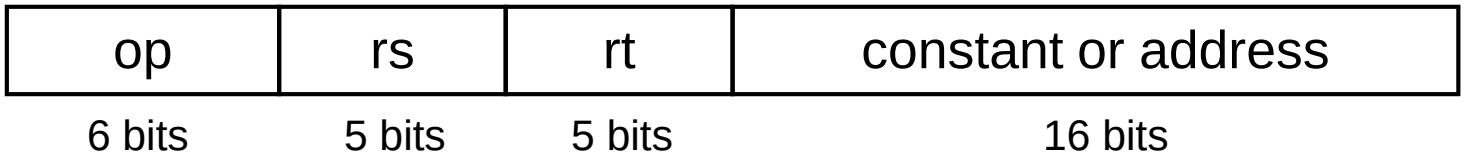



```
add $t0, $s1, $s2
```

0	17	18	8	0	32
000000	10001	10010	01000	00000	100000

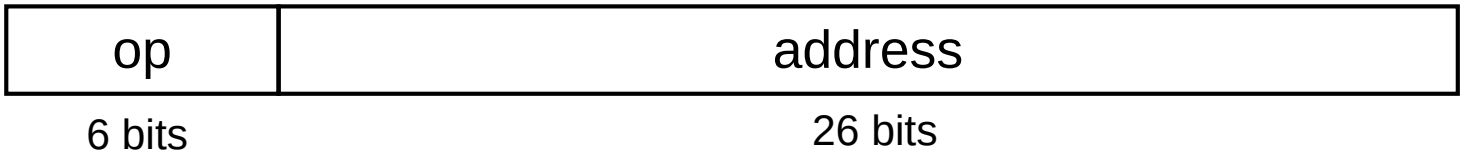
$00000010001100100100000000100000_2 = 02324020_{16}$

MIPS I-format Instructions



- Immediate arithmetic
  - `addi $s1, $s2, 20`
  - rs/rt: source/destination register number
  - Constant: –2^15 to +2^15 – 1
- Load/store instructions
  - `lw $t0, 32($s3)`
  - rs/rt: source/destination register number
  - Address: offset added to base address in rs

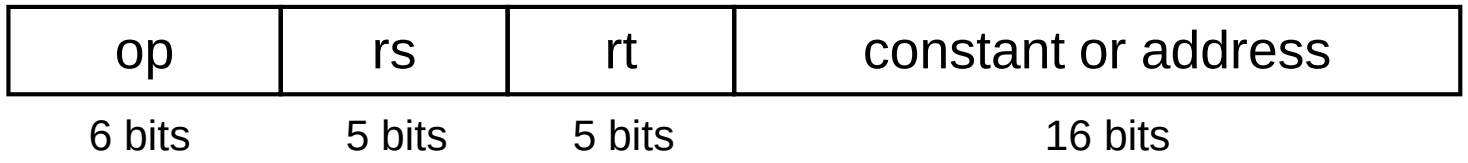
MIPS J-format Instructions



- Jump (j) targets could be anywhere in text segment
  - `j L1`
  - Encode full address in instruction
- (Pseudo)Direct jump addressing
  - Target address = $PC_{31…28}$: (address × 4)

Branch Addressing

- Branch instructions specify
  - `beq rs, rt, L1`
  - Opcode, two registers, target address
- Most branch targets are near branch
  - Forward or backward



- PC-relative addressing
  - Target address = PC + offset × 4
  - PC already incremented by 4 by this time

Addressing Mode Summary