CS61C
Virtual Memory Wrap-Up + Processor Datapath

Lecture 20

April 9, 1999

Dave Patterson
(http.cs.berkeley.edu/~patterson)

www-inst.eecs.berkeley.edu/~cs61c/schedule.html

Outline

- Review Virtual Memory
- Introduce Datapath Top-Down
- Basic Components and HW Building Blocks
- Administrivia, "Computers in the News"
- Designing an Arithmetic Logic Unit (ALU)
- 1-bit ALU
- 32-bit ALU
- Conclusion

Review 1/2

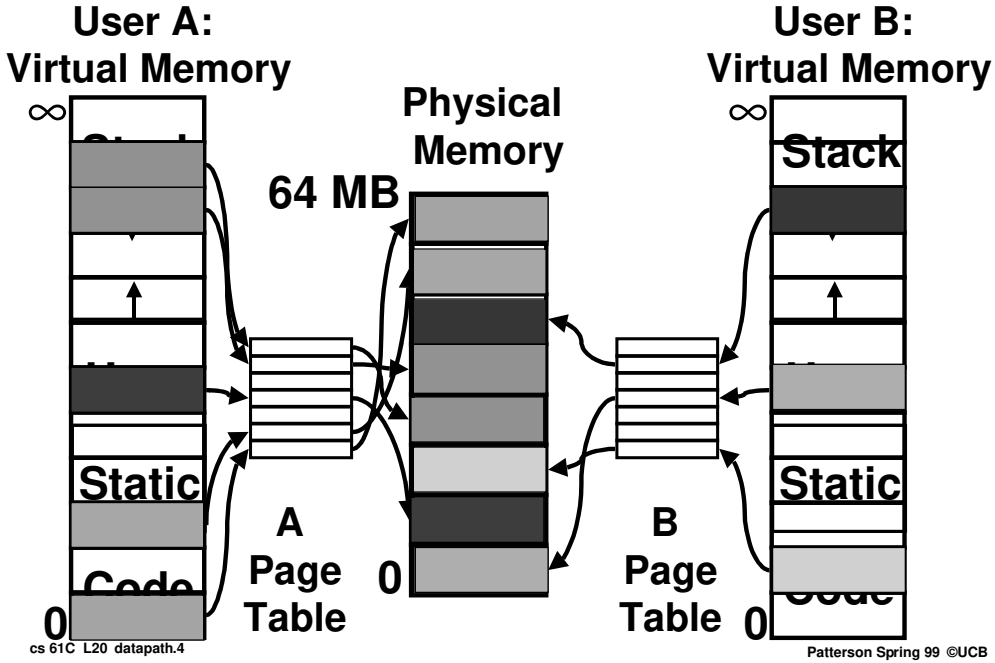
- Virtual Memory allows protected sharing of memory between processes with less swapping to disk, less fragmentation than always swap or base/bound
- 3 Problems:

1) Not enough memory: Spatial Locality means small Working Set of pages OK

2) TLB to reduce performance cost of VM

3) Need more compact representation to reduce memory size cost of simple 1-level page table, especially for 64-bit address (See CS 162)

Review 2/2: Paging/Virtual Memory



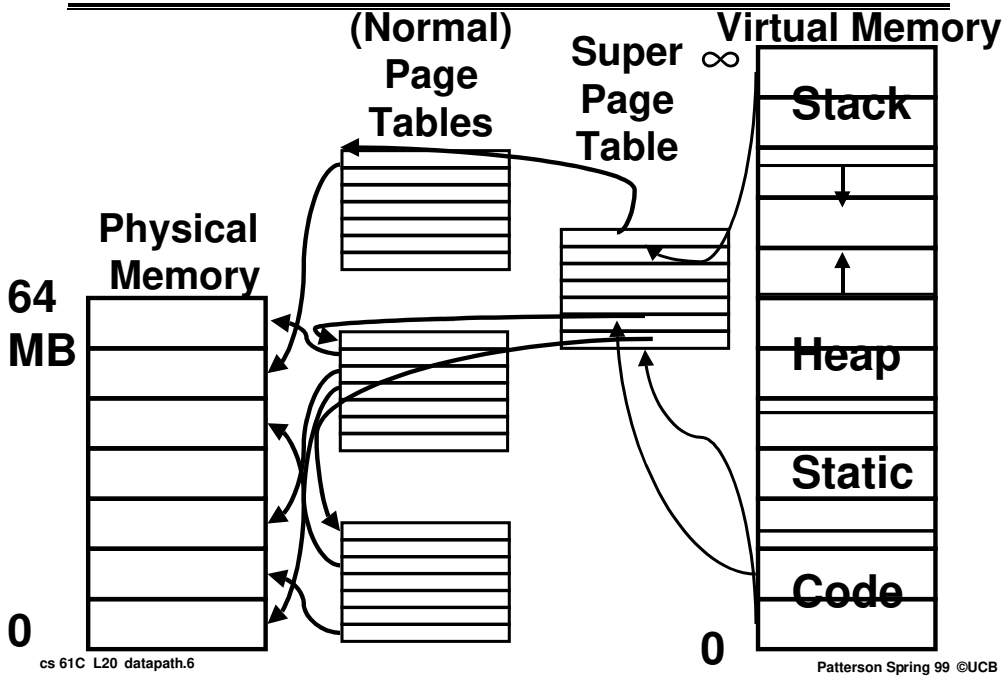
Reduce Page Table Space:

- Multilevel Page Table

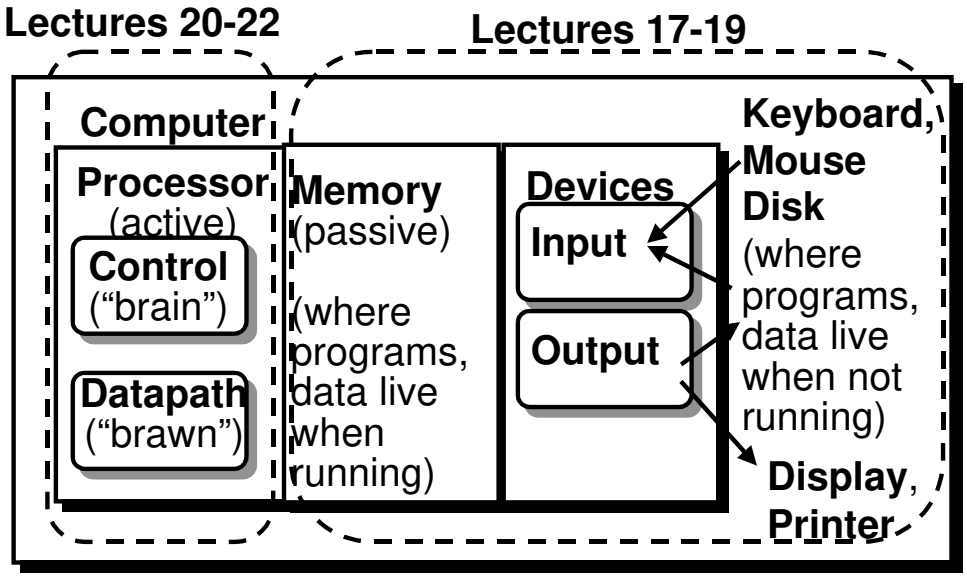
Super Page No.	Page Number	Offset
10 bits	10 bits	12 bits

- Super Pages map 2^{22} bytes (4 MB)
- Each Super Page Page Table Entry in Super Page Table points to a separate (normal) Page Table which maps 4MB into 1024 4KB (2^{12}) pages
- Save space by avoiding normal Page Table when no entry in Super Page Table

2-level Page Table



Anatomy: 5 components of any Computer



Deriving the Datapath for a MIPS Processor

- Start with instruction subset in 3 instruction classes to derive datapath

Memory-reference: `lw`, `sw`

Arithmetic-logical: `add`, `sub`, `and`, `or`

Branch: `beq`

- This subset illustrates shows most of the difficult steps in executing instructions

Up to 5 Steps in Executing MIPS Subset

- All instructions have common first two steps:

1) Fetch Instruction and Increment PC (Memory[PC]; PC = PC + 4)

2) Read 1 or 2 Registers (`lw` reads 1 reg)

Up to 5 Steps in Executing MIPS Subset

- 3rd step depends on instruction class

3) for Memory-reference: Calculate Address (Address = Reg[rs]+Imm)

3) for Arithmetic-logical: Calculate Result (Result = Reg[rs] op Reg[rt], op is +,-,&,|)

3) for Branch: Compare (equal = (Reg[rs] == Reg[rt]))

Up to 5 Steps in Executing MIPS Subset

- 4th step depends on instruction class

4) for `lw`: Fetch Data in Memory (Data = Memory[Address])

4) for `sw`: Memory[Address] = Reg[rt]

4) for Arithmetic-logical: Write Result (Reg[rd] = Result)

4) for Branch: Compare (if (Equal) PC = PC + Imm)

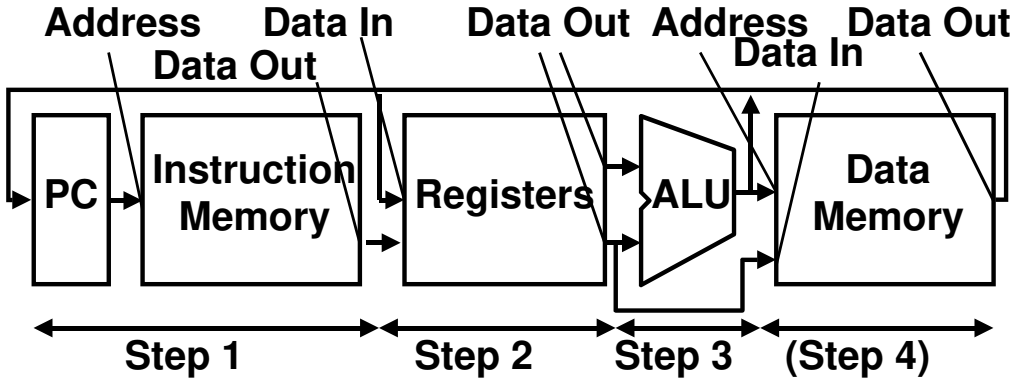
- 5th step only for `lw`; rest are done

5) for `lw`: Write Result (Reg[rt] = Data)

What is needed for Datapath from 5 steps

- PC
- 32 Registers
- Unit to perform +,-, &, |
  - Called an Arithmetic-Logic Unit, or ALU
- Memory for Instructions, Data
- Some miscellaneous registers to hold results between steps: Address, Data, Equal

Putting Together a Datapath for MIPS



- How can have separate Instruction Memory and Data Memory?
- Separate Caches for Instructions and for Data

Administrivia

- Project 5: Due 4/14: design and implement a cache (in software) and plug into instruction simulator
- Next Readings: 5.1 (skip logic, clocking), 5.2, 4.5 (pages 230-236), 4.6 (pages 250-253, 264; skim 254-257), 4.7 (pages 265-268, 273; skim 269-271)
  - How many lectures to cover: 2?
- 9th homework: Due Friday 4/16 7PM
  - Exercises 7.35, 4.24

Administrivia: Courses for Telebears

- Take courses from great teachers!
- Top Faculty / Course (may teach soon)

• CS 150 logic design	Katz	6.2	F92
• CS 152 computer HW	Patterson	6.7	S95
• CS 164 compilers	Rowe	6.1	S98
• CS 169 SW engin.	Brewer	6.2	S98
• CS 174 combinatorics	Sinclair	6.1	F97
• CS 186 data bases	Wang	6.2	S98
• EE 130 IC Devices	Hu	6.2	S97
• EE141 Digital IC Design	Rabaey	6.3	S97

hkn.eecs/toplevel/coursesurveys.html

"Computer (Technology) in the News"

- "A Milestone on the Road to Ultrafast Computers", N.Y. Times, April 6, 1999
- tunneling magnetic junction random access memory (tmj-ram) by IBM researchers
- A new kind of memory that could fundamentally alter computer design early in the next century... combine the best features of computer disks ... and memory chips... (No hierarchy: fast as cache, dense as disk)
- a crucial step toward new class of materials and microelectronics-- "spintronics"--based on ability to detect and control spins of electrons in ferromagnetic materials

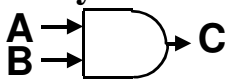
Contructing the Datapath Components

- Instruction Memory and Data Memory are just caches, as seen before
- PC, 32 Registers built from hardware called "registers" which each store 1 word
- Leaves ALU for MIPS subset
- (For full MIPS instruction set, need multiply, divide: do that later)
- First describe Hardware Building Blocks

Hardware Building Blocks (for ALU)

AND Gate

Symbol

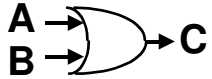


Definition

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate

Symbol

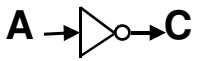


Definition

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Inverter

Symbol

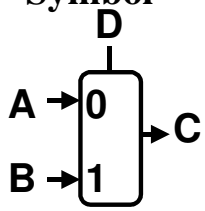


Definition

A	C
0	1
1	0

Multiplexor

Symbol

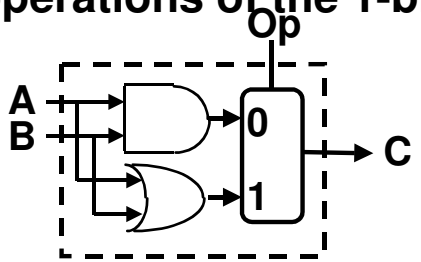


Definition

D	C
0	A
1	B

Arithmetic Logic Unit (ALU)

- MIPS ALU is 32 bits wide
- Start with 1-bit ALU, then connect 32 1-bit ALUs to form a 32-bit ALU
- Since hardware building block includes an AND gate and an OR gate, and since AND and OR are two of the operations of the 1-bit ALU, start here:

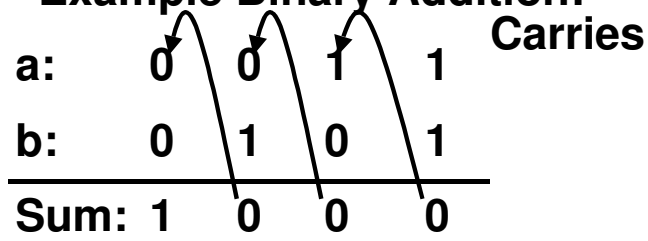


Definition

Op	C
0	A and B
1	A or B

What about Addition?

- Example Binary Addition:

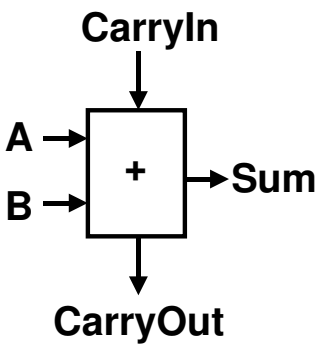


- Thus for any bit of addition:
  - The inputs are a_i, b_i, $CarryIn_i$
  - The outputs are Sum_i, $CarryOut_i$
- Note: $CarryIn_{i+1} = CarryOut_i$

1-Bit Adder

"Full Adder"

Symbol



Definition

A	B	CarryIn	CarryOut	Sum
0	0	0	0	0
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Constructing Hardware to Match Definition

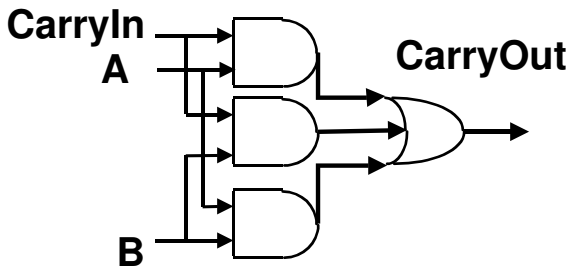
- Given any table of binary inputs for a binary output, programs can automatically connect a minimal number of AND gates, OR gates, and Inverters to produce the desired function
- Such programs generically called "Computer Aided Design", or CAD

Example: HW gates for CarryOut

- Values of Inputs when CarryOut is 1:

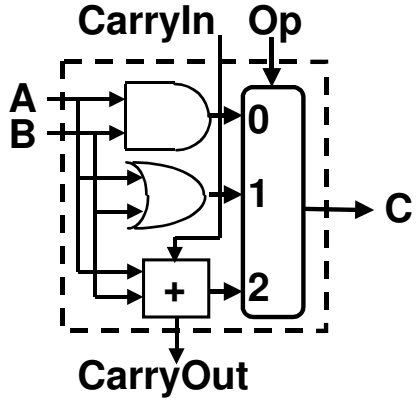
A	B	CarryIn
0	1	1
1	0	1
1	1	0
1	1	1

- Gates for CarryOut signal:



- Gates for Sum left as exercise to Reader

Add 1-bit Adder to 1-bit ALU

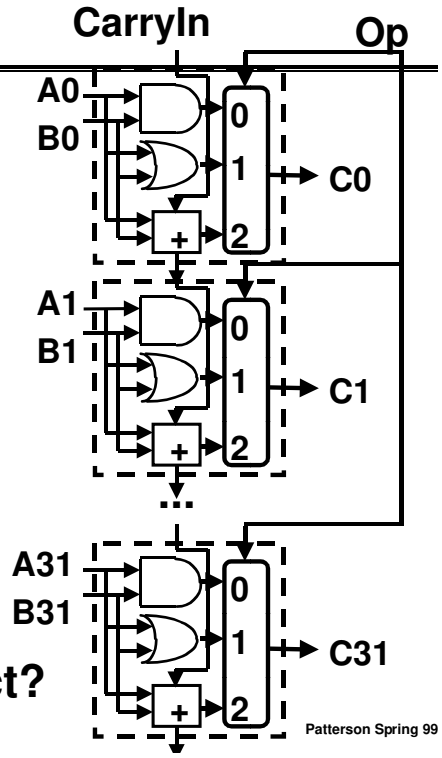


Definition

Op	C
0	A and B
1	A or B
2	A + B + CarryIn

- Now connect 32 1-bit ALUs together

32-bit ALU



- Connect $CarryOut_i$ to $CarryIn_{i+1}$
- Connect 32 1-bit ALUs together
- Connect Op to all 32 bits of ALU
- Does 32-bit And, Or, Add
- What about subtract?

Patterson Spring 99 ©UCB

2's comp. shortcut: Negation (Lecture 7)

- Invert every 0 to 1 and every 1 to 0, then add 1 to the result
  - Sum of number and its inverted rep. ("one's complement") must be $111...111_{two}$
  - $111...111_{two} = -1_{ten}$
  - Let x' mean the inverted representation of x
  - Then $x + x' = -1 \Rightarrow x + x' + 1 = 0 \Rightarrow x' + 1 = -x$
- Example: -4 to +4 to -4

$x: \ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1100_{two}$
$x': \ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0011_{two}$
$+1: \ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0100_{two}$
$()': \ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1011_{two}$
$+1: \ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1100_{two}$

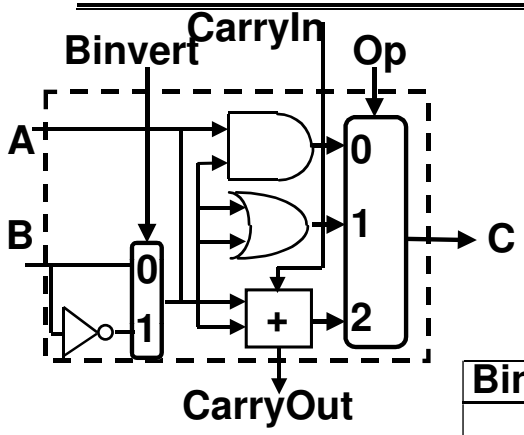
Patterson Spring 99 ©UCB

How Do Subtract?

- Suppose added input to 1-bit ALU that gave the one's complement of B
- What happens if set CarryIn0 to 1 in 32-bit ALU?
- Sum = A + B + 1
- Then if select inverted B ($\overline{B}$), Sum is $A + \overline{B} + 1 = A + (\overline{B} + 1) = A + (-B) = A - B$
- Therefore can do subtract as well as And, Or, Add if modify 1-bit ALU

Patterson Spring 99 ©UCB

1-bit ALU with Subtract Support



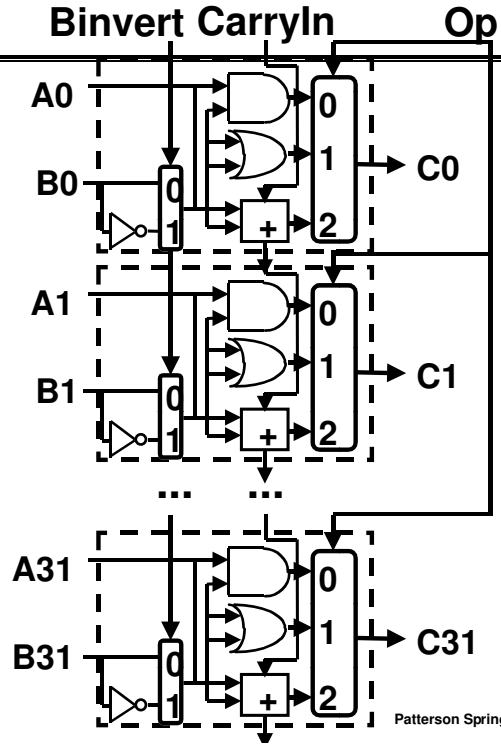
Definition

Binvert	Op	C
0	0	A and $\overline{B}$
1	0	A and $\overline{B}$
0	1	A or B
1	1	A or $\overline{B}$
0	1	A + $\overline{B}$ + CarryIn
1	1	A + $\overline{B}$ + CarryIn

Patterson Spring 99 ©UCB

32-bit ALU

- 32-bit ALU made from AND gates, OR gates, Inverters, Multiplexors
- Performs 32-bit AND, OR, Addition, Subtract (2's complement)



Patterson Spring 99 ©UCB

"And in Conclusion.." 1/1

- Virtual Memory shares physical memory between several processes via paging
- Datapath components visible in the instruction set: PC, Registers, Memory, ALU
- Hardware building blocks: And gate, Or gate, Inverter, Multiplexor
- Build Adder via Abstraction: decompose into 1-bit ALUs
- Seen how a computers adds, subtracts
- Next: How a computer Multiplies, Divides

Patterson Spring 99 ©UCB