CSE 2021
COMPUTER ORGANIZATION

HUGH CHESSER
CSE B 1012U

W2-W

From Last time…

$$\text{MIPS} = \frac{\text{Instruction Count for program}}{\text{Execution time for program} \times 10^6}$$

$$= \frac{\text{Ins Count}}{\text{Ins Count} \times \text{CPI} \times \text{Cyle period} \times 10^6} = \frac{\text{Clock Rate}}{\text{CPI} \times 10^6}$$

Activity 6: For a CPU, instructions from a high-level language are classified in 3 classes

Instruction Class	A	B	C
CPI for the instruction class	1	2	3

Two SW implementations with the following instruction counts are being considered

	Instruction counts (in billions) for each instruction class		
	A	B	C
Implementation 1	5	1	1
Implementation 2	10	1	1

Assuming that the clock rate is 500 MHz, calculate the (a) MIPS and (b) execution time

Implementation 1 - (a) 350 mips, (b) 20 seconds.

Exercise

The following table shows results for the SPEC2006 benchmark programs on an AMD Barcelona.

Benchmark name	Instr. Count (billions)	Execution time (seconds)	Reference time (seconds)
Perl	2118	500	9770
mcf	336	1200	9120

Calculate: (a) CPI if the clock cycle time is 0.333 ns

(b) SPECratio

(c) Geometric mean of the SPECratio

(d) MIPS

MIPS Instruction Set I

Topics:

- Arithmetic Instructions
- Registers, Memory, and Addressing
- Load and Save Instructions
- Signed and Unsigned Numbers
- Logical Operations
- Instructions for making decisions (Branch Instructions)

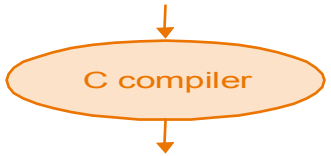
- Patterson: Today Sections 2.1 – 2.3.
  - Monday – 2.4, 2.6

Levels of Programming

1. Recall that a CPU can only understand binary machine language program
2. Writing binary machine language program is cumbersome
3. An intermediate solution is to write assembly language program that can easily be translated (assembled) to binary language programs
4. In this course we will cover MIPS ISA used by NEC, Nintendo, Silicon Graphics, and Sony
5. MIPS is more primitive than higher level languages with a very restrictive set of instructions

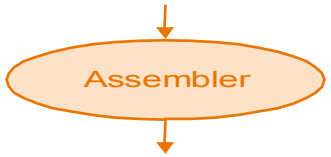
High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```



Assembly language program (for MIPS)

```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```

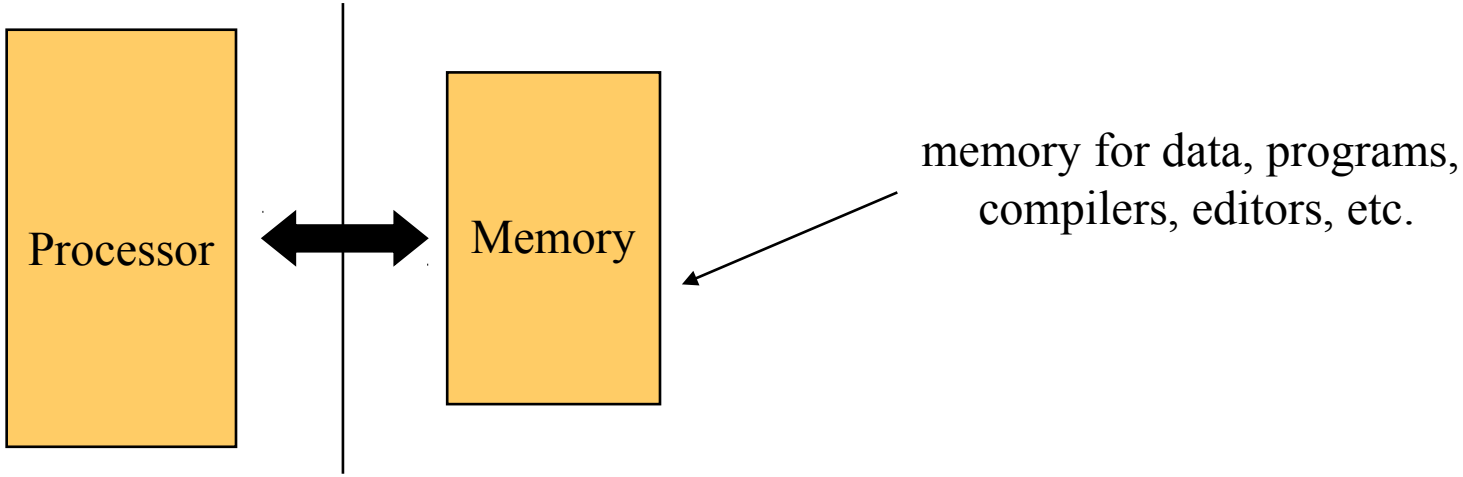


Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000100011100001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Fetch and Execute

1. Instructions are stored in the form of bits
2. Programs are stored in memory and are read or written just like data



3. Fetch & Execute Cycle
   - Instructions are fetched and put into a special register
   - Bits in the register "control" the subsequent actions
   - Data if required is fetched from the memory and placed in other registers
   - Fetch the “next” instruction and continue

Addition & Subtraction

Category	Instruction	Example	Meaning	Comments
Arithmetic	add	`add $s1,$s2,$s3`	`$s1 ← $s2+$s3`	overflow detect
	subtract	`sub $s1,$s2,$s3`	`$s1 ← $s2-$s3`	overflow detect

	$s0	$s1	$s2	$s3	$s4	$s5	$s6	$s7
$s0 - $s7		g	h	i	j			

Example:

C: f = (g + h) – (i + j);

MIPS Code:

Step 1: Specify registers containing variables

	$t0	$t1	$t2	$t3	$t4	$t5	$t6	$t7	$t8	$t9
$t0 - $t7	g+h	i+j	final							

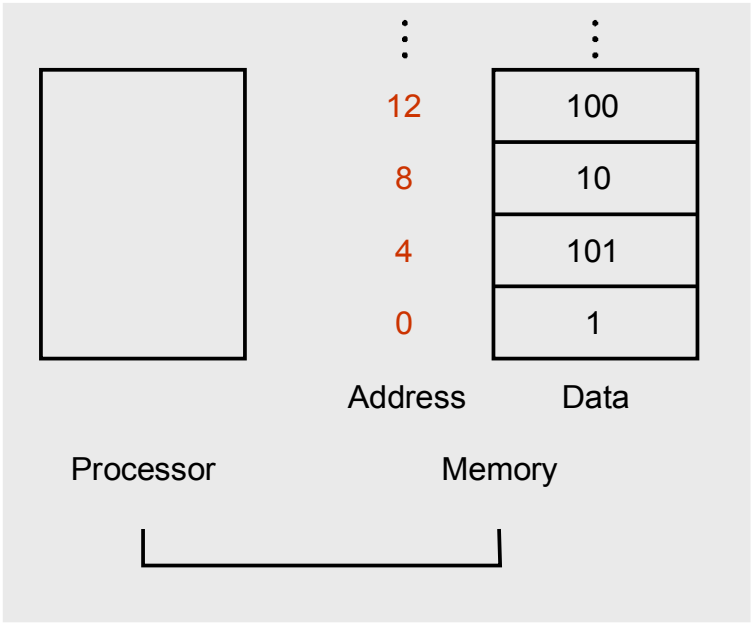
Step 2: Express instruction in MIPS

MIPS code:

```
add $t0,$s1,$s2      # $t0 ← $s1 + $s2
add $t1,$s3,$s4      # $t1 ← $s3 + $s4
sub $t2,$t0,$t1      # $t2 ← $t0 - $t1
```

Memory Organization

1. Memory can be viewed as a large one dimensional array of cells
2. To access a cell, its address is required (Addresses are indices to the array)
3. In MIPS, each cell is 1 word (4 bytes) long
4. Each word in a memory has an address, which is a multiple of 4
5. Length of an address is 32 bits, hence
   minimum value of address = 0
   maximum value of address = $(2^{32} - 1)$
6. Data is transferred from memory into registers using data transfer instructions



Category	Instruction	Example	Meaning	Comments
Data transfer	load word	`lw $s1,100($s2)`	`$s1 ← memory[$s2+100]`	Memory to Register
	store word	`sw $s1,100($s2)`	`memory[$s2+100]← $s1`	Register to memory

Data Transfer Instructions

Category	Instruction	Example	Meaning	Comments
Data transfer	load word	lw $s1,100($s2)	$s1 ← memory[$s2+100]	Memory to Register
	store word	sw $s1,100($s2)	memory[$s2+100]← $s2	Register to memory

Example: C instruction: g = h + A[k]

Register Allocation:

$s1 contains computed value of g; $s2 contains value of h

$s3 contains base address of array (address of A[0])

$s4 contains value of k;

MIPS Code:

```
add $t1,$s4,$s4        # $t1 = 2 x k
add $t1,$t1,$t1        # $t1 = 4 x k
add $t1,$t1,$s3        # $t1 = address of A[0] + 4 x k
lw  $t0,0($t1)         # $t0 = A[k]
add $s1,$s2,$t0        # $s3 = h + A[k]
```

So far we have learned …

MIPS

— loading words but addressing bytes

— addition and subtraction operations on registers only

Instructions	Meaning
`add $s1,$s2,$s3`	`# $s1 = $s2 + $s3 (arithmetic)`
`sub $s1,$s2,$s3`	`# $s1 = $s2 – $s3 (arithmetic)`
`lw $s1,100($s2)`	`# $s1 = Memory[$s2+100] (data transfer)`
`sw $s1,100($s2)`	`# Memory[$s2+100] = $s1 (data transfer)`

Activity 1: Write the MIPS assembly code for the following C assignment instruction

A[12] = h + A[8]

assuming that the variable h is stored in $s2 and the base address of the array A is in $s3.

MIPS to Binary Machine Language (1)

Example: `add $t0,$s1,$s2`

Binary Machine Language Equivalent:

000000 10001 10010 01000 00000 100000

Can we derive the binary machine language code from the MIPS instruction?

MIPS field for arithmetic instructions:

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
opcode	1st operand	2nd operand	destination	shift	function

Registers

1. Registers are memory cells
2. In MIPS, data must be in registers before arithmetic operations can be performed
3. Size of each register is 32 bits, referred to as a word (1 word = 4 bytes = 32 bits)
4. MIPS has a total of 32 registers

Name	Register number	Usage
$zero	0	Constant value of 0
$v0-$v1	2 - 3	Values for results and expression evaluation
$a0-$a3	4 - 7	Input arguments to a procedure
$t0-$t7	8 - 15	Not preserved across procedures (temp)
$s0-$s7	16 - 23	Preserved across procedure calls
$t8-$t9	24 - 25	More temporary registers
$gp	28	Global pointer
$sp	29	Stack pointer, points to last location of stack
$fp	30	Frame pointer
$ra	31	Return address from a procedure call

Representing MIPS Instructions

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
opcode	1st operand	2nd operand	destination	shift	function

For arithmetic operations (R):

- Opcode (op) = 0
- Function (funct) = 32 for add, 34 for sub

Example: `add $t0,$s1,$s2` (Values of Registers: $t0 = 9, $s1 = 17, $s2 = 18)

op = $0_{10} = (000000)_2$

rs = $17_{10} = (10001)_2$

rt = $18_{10} = (10010)_2$

rd = $8_{10} = (01000)_2$

shamt is not used = $(00000)_2$

funct = $32_{10} = (100000)_2$

leads to the binary machine language code: 000000 10001 10010 01000 00000 100000

MIPS Fields for Data Transfer Operations

op	rs	rt	address
6 bits	5 bits	5 bits	16 bits
opcode	1st operand	2nd operand	Memory address (offset)

For data transfer operations (I):

- Opcode (op) = 35 for load (lw) and 43 for save (sw)

Example: `lw $t0,32($s3)` # (Values of Registers: $t0 = 9, $s3 = 19)

op = $35_{10} = (100011)_2$

rs = $19_{10} = (10011)_2$

rt = $8_{10} = (01000)_2$

address = $32_{10} = (0000\ 0000\ 0010\ 0000)_2$

leads to the binary machine language code: 100011 10011 01000 0000000000100000

Example

Activity 2: Consider the C instruction

A[300] = h + A[300]

A. Write the equivalent MIPS code for the above C instruction assuming $t1 contains the base address of array A (i.e., address of A[0]) and $s2 contains the value of h

B. Write the binary machine language code for the result in part A.