CS 61C Fall 2014 Discussion 2 – MIPS

C	MIPS

`// $s0 -> a, $s1 -> b`
`// $s2 -> c, $s3 -> z`
 	
`int a = 4, b = 5, c = 6, z;`
`z = a + b + c + 10;`
	`addiu $s0, $0, 4`
	`addiu $s1, $0, 5`
	`addiu $s2, $0, 6`
	`addu $s3, $s0, $s1`
	`addu $s3, $s3, $s2`
	`addiu $s3, $s3, 10`

`// $s0 -> int * p = intArr;`
`// $s1 -> a;`
 	
`*p = 0;`
`int a = 2;`
`p[1] = p[a] = a;`
	`sw $0, 0($s0)`
	`addiu $s1, $0, 2`
	`sw $s1, 4($s0)`
	`sll $t0, $s1, 2`
	`add $t0, $t0, $s0`
	`sw $s1, 0($t0)`

`// $s0 -> a, $s1 -> b`
 	
`int a = 5, b = 10;`
`if(a + a == b) {`
`  a = 0;`
`} else {`
`  b = a - 1;`
`}`
	`addiu $s0, $0, 5`
	`addiu $s1, $0, 10`
	`addu $t0, $s0, $s0`
	`bne $t0, $s1, else`
	`xor $s0, $0, $0`
	`j exit`
	`else:`
	`addiu $s1, $s0, -1`
	`exit:`

`// computes s1 = 2^30`
 	
`s1 = 1;`
`for(s0=0;s0<30;s++) {`
`  s1 *= 2;`
`}`
	`addiu $s0, $0, 0`
	`addiu $s1, $0, 1`
	`addiu $t0, $0, 30`
	`loop:`
	`beq $s0, $t0, exit`
	`addu $s1, $s1, $s1`
	`addiu $s0, $s0, 1`
	`j loop`
	`exit:`

`int sum(int n) {`
`  int sum;`
`  for(sum=0;n>0;sum+=n--);`
`  return sum;`
`}`
	`sum:`
	`xor $v0, $0, $0`
	`loop:`
	`blez $a0, exit`
	`addu $v0, $v0, $a0`
	`addiu $a0, $a0, -1`
	`j loop`
	`exit:`
	`jr $ra`

Implement `streq`, which sets `$v0` to true only when its two character pointer arguments (`$a0` and `$a1`) point to equal strings, first in C, then in MIPS.

C	MIPS

`int streq(char * s1, char * s2) {`
`  do {`
`  if(*s1 != *s2) {`
`  return 0;`
`  }`
`  s2++;`
`  } while(*s1++);`
`  return 1;`
`}`
	`streq:`
	`  lb $t0, 0($a0)`
	`  lb $t1, 0($a1)`
	`  bne $t0, $t1, false`
	`  beq $t0, $0, true`
	`  addiu $a0, $a0, 1`
	`  addiu $a1, $a1, 1`
	`  j streq`
	`false:`
	`  xor $v0, $0, 0`
	`  jr $ra`
	`true:`
	`  addiu $v0, $0, 1`
	`  jr $ra`

What are the instructions to branch on each of the following conditions?

`$s0 < $s1`	`$s0 <= $s1`	`$s0 > 1`	`$s0 >= 1`
`slt $t0, $s0, $s1` `bne $t0, $0, label`	`slt $t0, $s1, $s0` `beq $t0, $0, label`	`sltiu $t0, $s0, 2` `beq $t0, $0, label`	`bgtz $s0, label`

There are a few different meanings that the term "unsigned" takes in MIPS. Describe all three meanings, and give example instructions.

	Unsigned Meaning	Examples
1	Do not sign extend loaded data	`lbu`, `lhu`
2	No errors on **signed** overflow	`addu`, `subu`, `addiu`
3	Perform unsigned operations	`multu`, `divu`, `sltu`, `sltiu`

What is the distinction between zero extension and sign extension?

Sign extension and zero extension both extend a data type of a smaller bit width M to a data type of a larger bit width N. To zero extend from M to N, the lower M bits are kept unchanged, while the top $N - M$ bits are 0. To sign extend from M to N, the lower M bits are again unchanged, but the top $N - M$ bits are filled with the bit value of the bit at location $M - 1$.

What is the purpose of sign extension, and when is zero extension used instead of sign extension in MIPS?

Sign extension preserves the sign of a data type undergoing extension. The instructions `andi`, `ori`, and `xori` specifically zero extend their immediate values. The instructions `lbu` and `lhu` specifically zero extend their data loaded from memory. Also, jump addresses and shift amounts are not sign-extended in MIPS but are not exactly zero extended either.