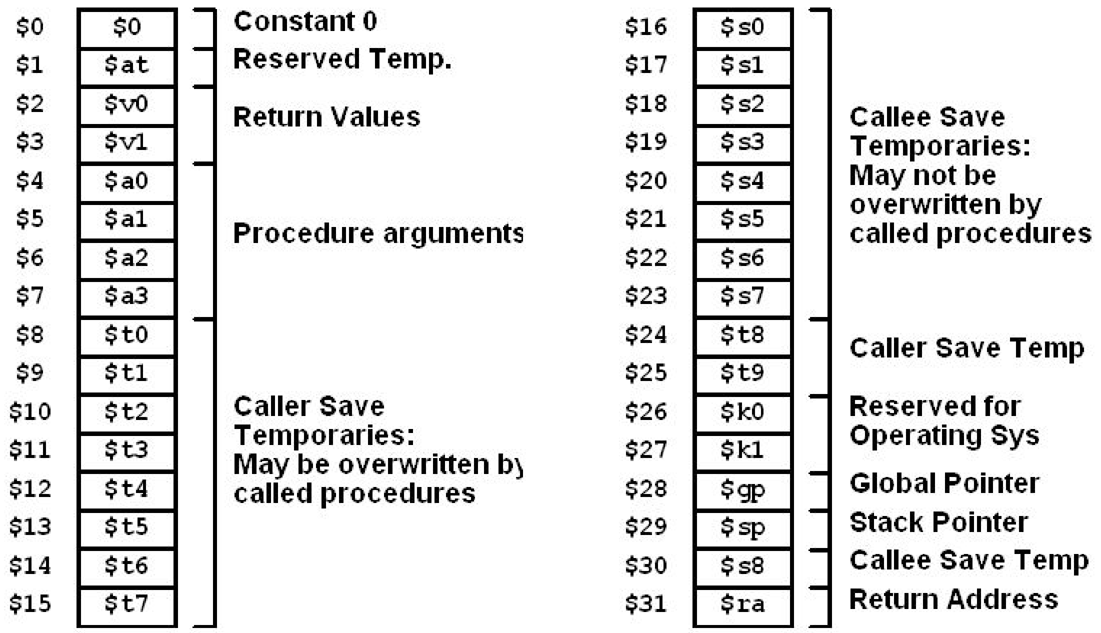
COMP1521 Tutorial 04

MIPS

- Assembly language
  - Reduced Instruction Set Computing
- There are many different kinds of assembly
  - X86
  - ARM
  - MIPS
- Simple set of instructions to learn assembly

MIPS Registers

- Extremely high speed memory
  - Faster than ram
  - Built into processor itself



$0	$0	Constant 0	$16	$s0	Callee Save Temporaries: May not be overwritten by called procedures
$1	$at	Reserved Temp.	$17	$s1	
$2	$v0	Return Values	$18	$s2	
$3	$v1		$19	$s3	
$4	$a0	Procedure arguments	$20	$s4	
$5	$a1		$21	$s5	
$6	$a2		$22	$s6	
$7	$a3		$23	$s7	
$8	$t0	Caller Save Temporaries: May be overwritten by called procedures	$24	$t8	Caller Save Temp
$9	$t1		$25	$t9	
$10	$t2		$26	$k0	Reserved for Operating Sys
$11	$t3		$27	$k1	
$12	$t4		$28	$gp	Global Pointer
$13	$t5		$29	$sp	Stack Pointer
$14	$t6		$30	$s8	Callee Save Temp
$15	$t7		$31	$ra	Return Address

MIPS Registers

For each of the registers below, give their symbolic name and describe their intended use:

$0

$1

$2

$4

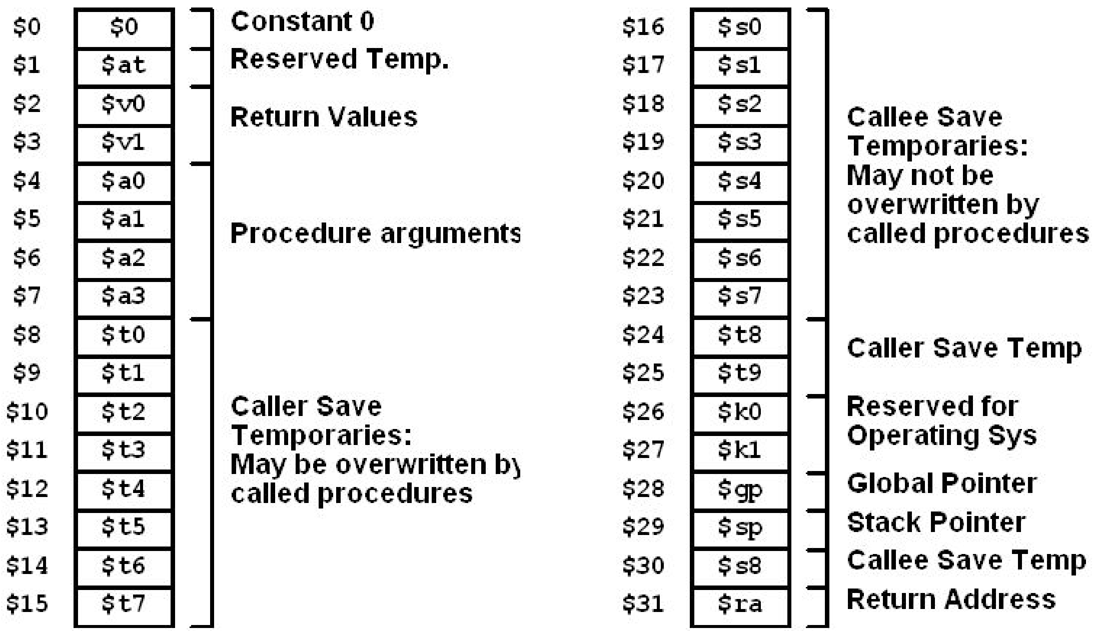
$8

$16

$26

$29

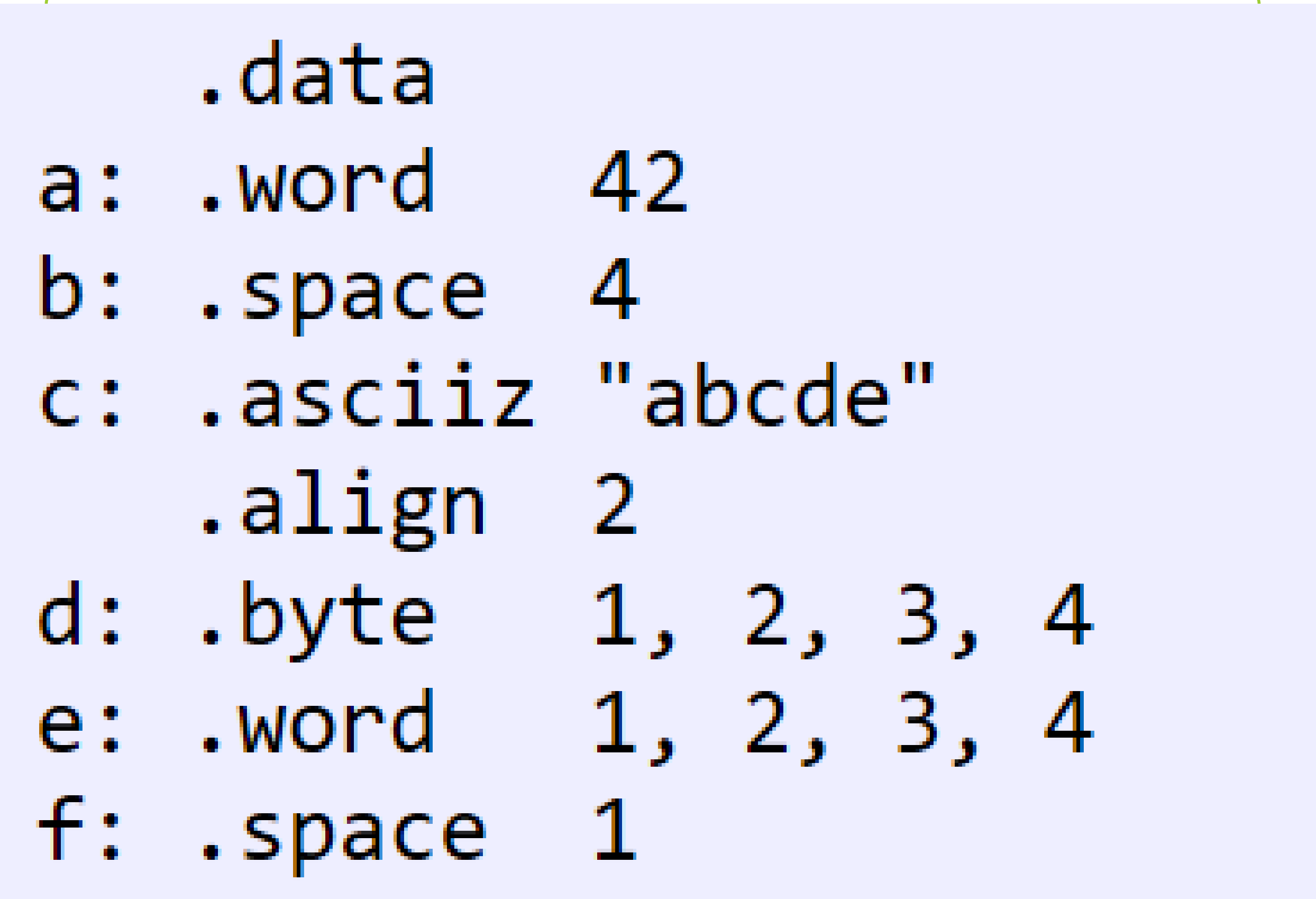
$31



$0	$0	Constant 0
$1	$at	Reserved Temp.
$2	$v0	Return Values
$3	$v1	
$4	$a0	Procedure arguments
$5	$a1	
$6	$a2	
$7	$a3	
$8	$t0	Caller Save Temporaries: May be overwritten by called procedures
$9	$t1	
$10	$t2	
$11	$t3	
$12	$t4	
$13	$t5	
$14	$t6	
$15	$t7	
$16	$s0	Callee Save Temporaries: May not be overwritten by called procedures
$17	$s1	
$18	$s2	
$19	$s3	
$20	$s4	
$21	$s5	
$22	$s6	
$23	$s7	
$24	$t8	Caller Save Temp
$25	$t9	
$26	$k0	Reserved for Operating Sys
$27	$k1	
$28	$gp	Global Pointer
$29	$sp	Stack Pointer
$30	$s8	Callee Save Temp
$31	$ra	Return Address

Memory in Assembly

If the data segment of a particular MIPS program starts at the address 0x10000020, then what addresses are the following labels associated with, and what value is stored in each 4-byte memory cell?



```
    .data
a:  .word   42
b:  .space  4
c:  .asciiz "abcde"
    .align  2
d:  .byte   1, 2, 3, 4
e:  .word   1, 2, 3, 4
f:  .space  1
```

MIPS Instructions

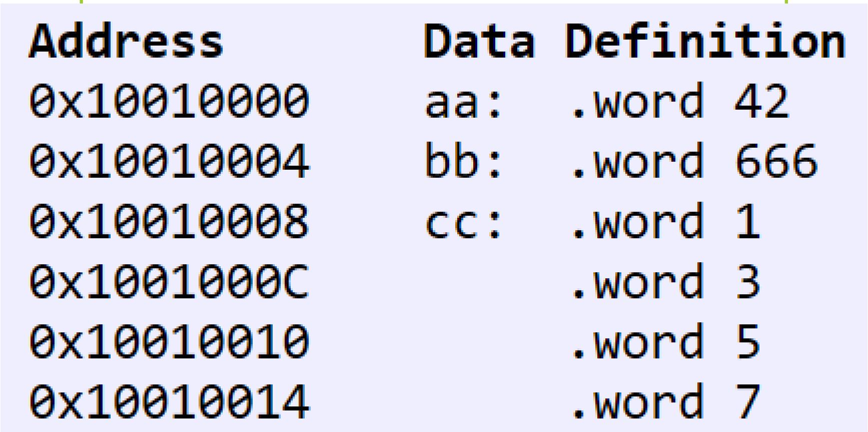
Give MIPS directives to represent the following variables:

- int u;
- int v = 42;
- char w;
- char x = 'a';
- double y;
- int z[20];

MIPS Instructions

What address will be calculated and what value will be loaded into register $t0 after each of the followng statements (or pairs of statements)?

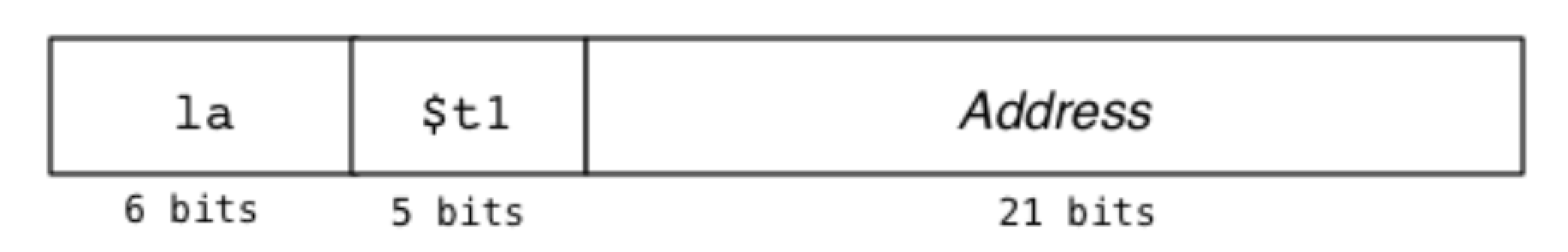
- la $t0, aa
- lw $t0, bb
- lb $t0, bb
- lw $t0, aa+4
- la $t1, cc
  lw $t0, ($t1)
- la $t1, cc
  lw $t0, 8($t1)
- li $t1, 8
  lw $t0, cc($t1)
- la $t1, cc
  lw $t0, 2($t1)



Address	Data	Definition
0x10010000	aa:	.word 42
0x10010004	bb:	.word 666
0x10010008	cc:	.word 1
0x1001000C		.word 3
0x10010010		.word 5
0x10010014		.word 7

How MIPS stores 32bit values

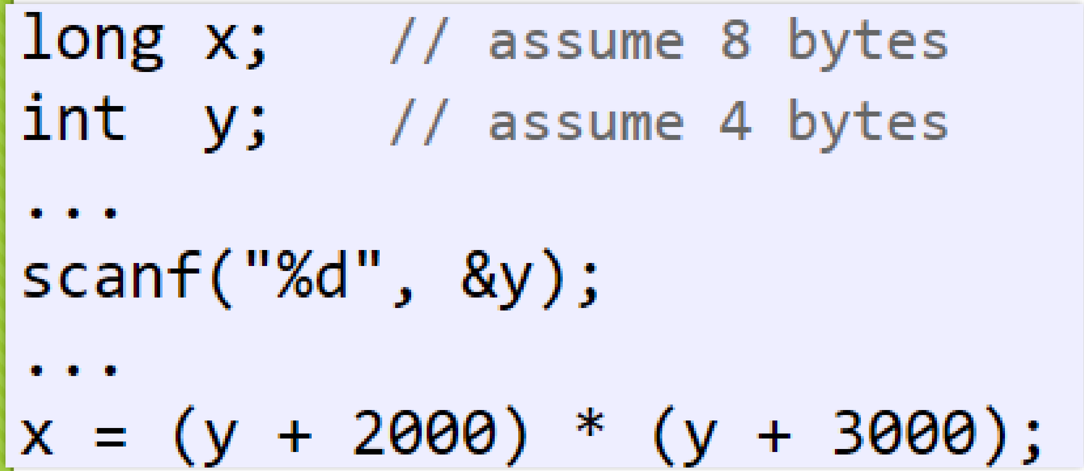
Each MIPS instruction is encoded in 32bits



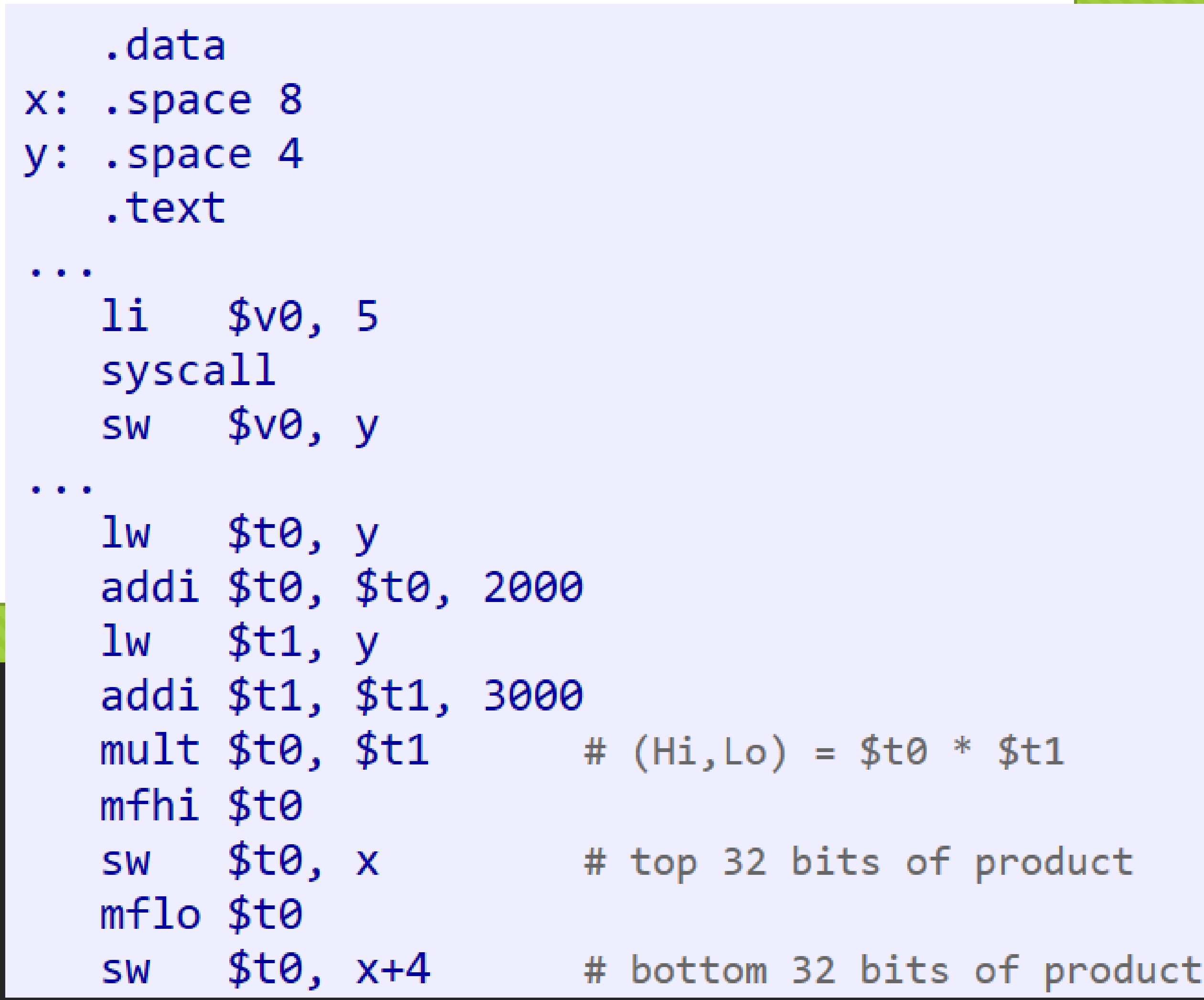
la	$t1	*Address*
6 bits	5 bits	21 bits

- Some instructions (la, li) takes 32bit arguments
- MIPS splits them into 16bit arguments and uses 2 *real* MIPS instructions to execute the instruction

C translated to MIPS

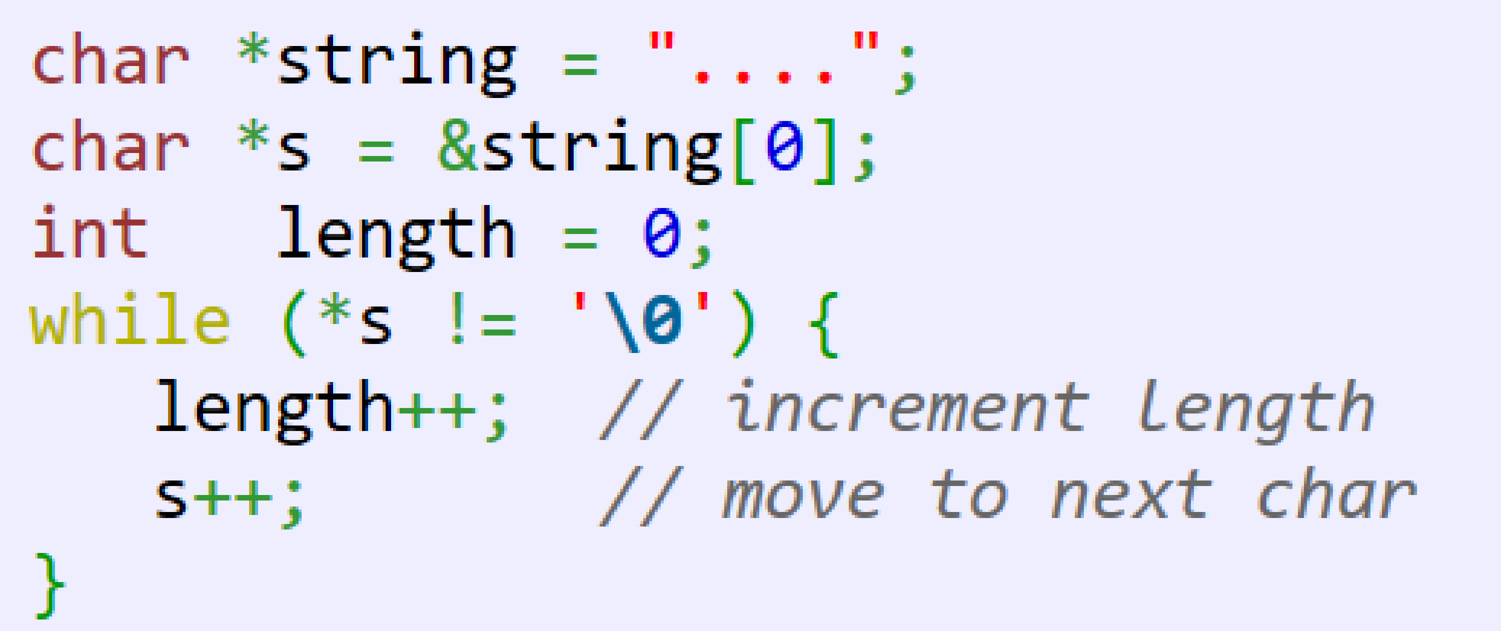


```c
long x;    // assume 8 bytes
int  y;    // assume 4 bytes
...
scanf("%d", &y);
...
x = (y + 2000) * (y + 3000);
```



```
    .data
x: .space 8
y: .space 4
    .text
...
    li   $v0, 5
    syscall
    sw   $v0, y
...
    lw   $t0, y
    addi $t0, $t0, 2000
    lw   $t1, y
    addi $t1, $t1, 3000
    mult $t0, $t1        # (Hi,Lo) = $t0 * $t1
    mfhi $t0
    sw   $t0, x          # top 32 bits of product
    mflo $t0
    sw   $t0, x+4        # bottom 32 bits of product
```

C to MIPS



```
char *string = "....";
char *s = &string[0];
int   length = 0;
while (*s != '\0') {
   length++;  // increment length
   s++;       // move to next char
}
```