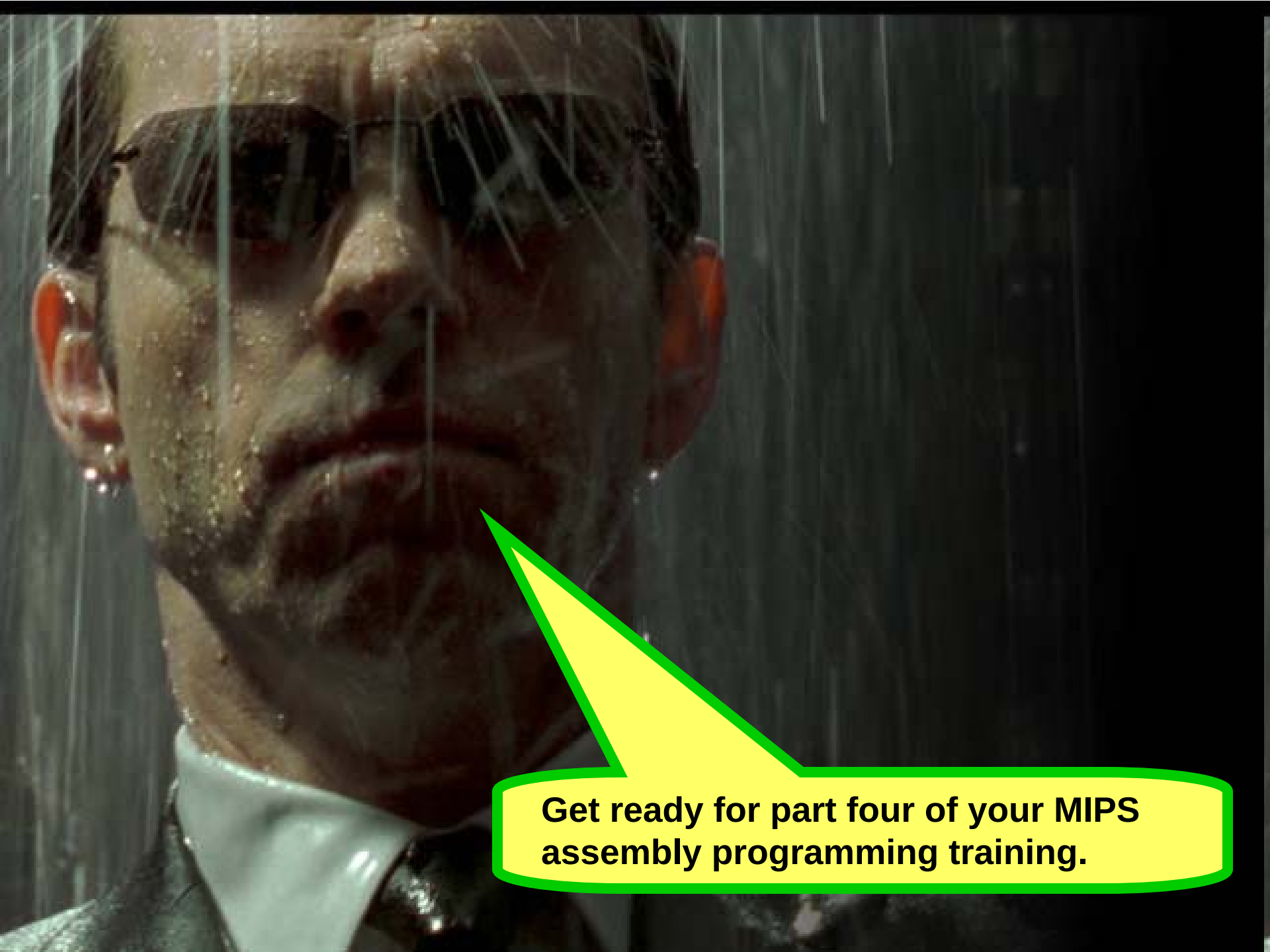
MIPS and SPIM tutorial

Part Four: Strings, Loops, If-Then-Else and Arrays

November 2008

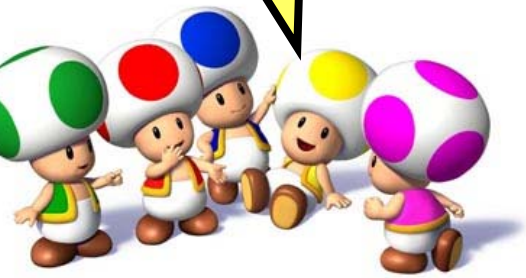
karl.marklund@it.uu.se


Get ready for part four of your MIPS assembly programming training.

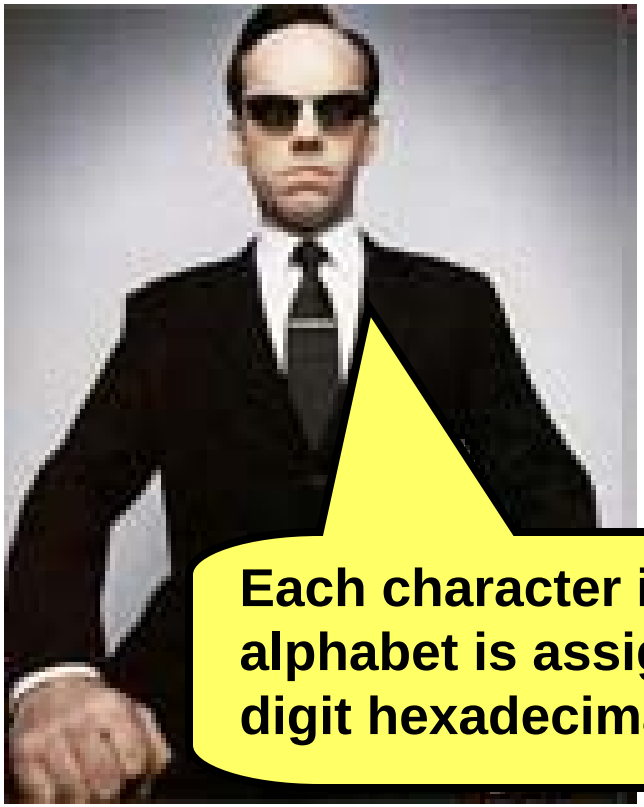
.asciiz

A directive that stores a NUL terminated string in the data segment.

Ok, but how is a string represented?



```
        .data

STR:    .asciiz  "abcdefghijklmnopqrstuvxyz"

        .text
        .globl   main

main:
```



Each character in the alphabet is assigned a 2 digit hexadecimal number.

ASCII NUL 0x00 is used to mark the end of a string.

Column: leftmost digit

Row: *rightmost digit*

Each character is assigned a 8 bit *ASCII value* – a byte.

The ASCII value of 'a' is *0x61*

The ASCII value of 'H' is *0x48*

ASCII:

American Standard Code for Information Interchange.

	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	[illegible]	[illegible]	[illegible]	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	[illegible]	[illegible]	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	del

```
        .data

STR:    .asciiz  "abcdefghijklmnopqrstuvxyz"

        .text
        .globl   main

main:   la       $t0, STR
        lw       $t1, 0($t0)

        lb       $t2, 0($t0)
        lb       $t3, 1($t0)
        lb       $t4, 2($t0)
        lb       $t5, 3($t0)

        jr       $ra
```

Load Byte: *lb*

Ok, just like Load Word but only loads one byte.

PCSpim

File Simulator Window Help

Open... Ctrl+O
Save Log File... Ctrl+S
Exit Alt+F4

Load the source file string.s

```
00   EPC      =                                    = 00000000
10   HI       =

R0  (r0) = 00000000  R8  (t0) =                                  = 00000000
R1  (at) = 00000000  R9  (t1) = 00000000  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000000  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000

[0x00400000]    0x8fa40000  lw $4, 0($29)              ; 175: lw $a0 0($sp)          # argc
[0x00400004]    0x27a50004  addiu $5, $29, 4           ; 176: addiu $a1 $sp 4        # argv
[0x00400008]    0x24a60004  addiu $6, $5, 4            ; 177: addiu $a2 $a1 4        # envp
[0x0040000c]    0x00041080  sll $2, $4, 2              ; 178: sll $v0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2            ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]      ; 180: jal main
[0x00400018]    0x00000000  nop                        ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10             ; 183: li $v0 10

        DATA
[0x10000000]...[0x10040000]     0x00000000

        STACK
[0x7fffeffc]                    0x00000000

        KERNEL DATA
[0x90000000]                    0x78452020  0x74706563  0x206e6f69  0x636f2000

Copyright 1990-2004 by James R. Larus (lar
All Rights Reserved.
DOS and Windows ports by David A. Carley (
Copyright 1997 by Morgan Kaufmann Publishe
See the file README for a full copyright n
Loaded: C:\Program Files\PCSpim\exceptions
C:\Documents and Settings\Karl Marklund\My                    2008\Tutorials By Karl Marklund\first_try
```

Single step until...

Open an assembly file

start 7 F.. xspi... 3 G.. Dar ... 7 M.. tmp... SV 10:06 PM

PCSpim

File Simulator Window Help

```
PC      = 00400028   EPC     = 00000000   Cause   = 00000000   BadVAddr= 00000000
Status  = 3000ff10   HI      = 00000000   LO      = 00000000

R0  (r0) = 00000000   R8  (t0) = 10010000   R16 (s0) = [illegible]
R1  (at) = [illegible]   R9  (t1) = [illegible]   R17 (s1) = 00000000   R25 (t9) = [illegible]
R2  (v0) = 00000000   R10 (t2) = 00000000   R18 (s2) = 00000000   R26 (k0) = 00000000
R3  (v1) = 00000000   R11 (t3) = 00000000   R19 (s3) = 00000000   R27 (k1) = 00000000
R4  (a0) = 00000000   R12 (t4) = 00000000   R20 (s4) = 00000000   R28 (gp) = 10008000
```

```
[0x00400000c]   0x00041080  sll $2, $4, 2                  ; 178: sll $v0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2                ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]          ; 180: jal main
[0x00400018]    0x00000000  nop                            ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10                 ; 183: li $v0 10
[0x00400020]    0x0000000c  syscall                        ; 184: syscall                # syscal
[0x00400024]    0x3c081001  lui $8, 4097 [STR]             ; 8: la     $t0, STR
[0x00400028]    0x8d090000  lw $9, 0($8)                   ; 9: lw     $t1, 0($t0)
```

```
        DATA
[0x10010000]..[0x10010000]   [illegible]
[0x10010000]                 0x64636261  [illegible]
[0x10010018]                 [illegible] 0x79787675
[0x1001002[illegible]  [0x10040000]   0x00000000

        STACK
[0x7fffeff[illegible]
```

```
[0x0040000[illegible]   lw $a0 0($sp)                      # argc
[0x0040000[illegible]   '6: [illegible]                    # argv
[0x0040000[illegible]   '7: ad[illegible]                  # envp
[0x0040000[illegible]   '8: sll
[0x0040001[illegible]   '9: addu
[0x0040001[illegible]   0: jal
[0x0040002[illegible]    la
```

$t0 is 10010000

The content at address 0x1001000 is 0x64636261

The label STR is refering to the address 0x1001000 in the data segment of the memory.

Single step again...

PCSpim

File Simulator Window Help

```
PC       = 0040002c   EPC      = 00000000   Cause    = 00000000   BadVAddr= 00000000
Status   = 3000ff10   HI       = 00000000   LO       = 00000000
                                General Registers
R0  (r0) = 00000000                          R16 (s0) = 00000000
R1  (at) = 00000000   R9  (t1) = 64636261    R17 (s1) = 00000000   R25 (t9) = 00000000
R2  (v0) = 00000000                          R18 (s2) = 00000000   R26 (k0) = 00000000
R3  (v1) = 00000000   R11 (t3) = 00000000    R19 (s3) = 00000000   R27 (k1) = 00000000
R4  (a0) = 00000000   R12 (t4) = 00000000    R20 (s4) = 00000000   R28 (gp) = 10008000
```

$t1 is 0x64636261

```
[0x00400010]    0x00c23021  addu $6, $6, $2          ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]    ; 180: jal main
[0x00400018]    0x00000000  nop                      ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10           ; 183: li $v0 10
[0x00400020]    0x0000000c  syscall                  ; 184: syscall                # syscal
[0x00400024]    0x3c081001  lui $8, 4097 [STR]       ; 8: la     $t0, STR
[0x00400028]    0x8d090000  lw $9, 0($8)             ; 9: lw     $t1, 0($t0)
[0x0040002c]    0x810a0000  lb $10, 0($8)            ; 11: lb    $t2, 0($t0)
```

```
        DATA
[0x10000000]...[0x10010000]
[0x10010000]                0x64636261
[0x10010010]                0x79787675
[0x10010020]...[0x10040000]  0x00000000

        STACK
[0x7fffeffc]                0x00000
```

The content at address 0x1001000 is 0x64636261

```
[0x00400004]    0x27a50004  addiu $5, $
[0x00400008]    0x24a60004  addiu $6, $
[0x0040000c]    0x00041080  sll $2, $4,
[0x00400010]    0x00c23021  addu $6, $6
[0x00400014]    0x0c100009  jal 0x00400
[0x00400024]    0x3c081001  lui $8, 409
[0x00400028]    0x8d090000  lw $9, 0($8
```

Single step again...

PCSpim

File Simulator Window Help

```
PC       = 00400030   EPC      = 00000000   Cause    = 00000000   BadVAddr= 00000000
Status   = 3000ff10   HI       = 00000000   LO       = 00000000
                                General Registers
R0  (r0) = 00000000   R8  (t0) = 10010000   R16 (s0) = 00000000
R1  (at) = 00000000   R9  (t1) = 64636261   R17 (s1) = 00000000   R25 (t9) = 00000000
R2  (v0) = 00000000   R10 (t2) = 00000061   R18 (s2) = 00000000   R26 (k0) = 00000000
R3  (v1) = 00000000   R11 (t3) = 00000000   R19 (s3) = 00000000   R27 (k1) = 00000000
R4  (a0) = 00000000   R12 (t4) = 00000000   R20 (s4) = 00000000   R28 (gp) = 10008000
```

```
[0x00400014]    0x0c100009  jal 0x00400024 [main]        ; 180: jal main
[0x00400018]    0x00000000  nop                           ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10                ; 183: li $v0 10
[0x00400020]    0x0000000c  syscall                       ; 184: syscall                    # syscal
[0x00400024]    0x3c081001  lui $8, 4097 [STR]            ; 8: la     $t0, STR
[0x00400028]    0x8d090000  lw $9, 0($8)                  ; 9: lw     $t1, 0($t0)
[0x0040002c]    0x810a0000  lb $10, 0($8)                 ; 11: lb    $t2, 0($t0)
[0x00400030]    0x810b0001  lb $11, 1($8)                 ; 12: lb    $t3, 1($t0)
```

```
        DATA
[0x10000000]...[0x10010000]   00000000
[0x10010000]                  0x64636261  0
[0x10010010]                  0x74737271  0x79787675
[0x10010020]...[0x10040000]   0x00000000

        STACK
[0x7fffeffc]                  0x00000
```

```
[0x00400008]    0x24a60004  addiu $6, $5, 4              7: addiu $a2 $a1 4                # envp
[0x0040000c]    0x00041080  sll $2, $4, 2                 8: sll
[0x00400010]    0x00c23021  addu $6, $6, $2               9: addu
[0x00400014]    0x0c100009  jal 0x00400024                10: jal
[0x00400024]    0x3c081001  lui $8, 4097                   la
[0x00400028]    0x8d090000  lw $9, 0($8                    lw
[0x0040002c]    0x810a0000  lb $10, 0($                    lb
```

$t2 is 0x00000061

Load Byte

The content at address 0x1001000 is 0x64636261

Single step again...

PCSpim

File Simulator Window Help

```
PC     = 00400034   EPC    = 00000000   Cause    = 00000000   BadVAddr= 00000000
Status = 3000ff10   HI     = 00000000   LO       = 00000000
                              General Registers
R0  (r0) = 00000000  R8  (t0) = 10010000  R16 (s0) = 00000000
R1  (at) = 00000000  R9  (t1) = 64636261  R17 (s1) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000061  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000062  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000

[0x00400018]    0x00000000  nop                          ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10               ; 183: li $v0 10
[0x00400020]    0x0000000c  syscall                      ; 184: syscall
[0x00400024]    0x3c081001  lui $8, 4097 [STR]           ; 8: la     $t0, STR
[0x00400028]    0x8d090000  lw $9, 0($8)                 ; 9: lw     $t1, 0($t0)
[0x0040002c]    0x810a0000  lb $10, 0($8)                ; 11: lb    $t2, 0($t0)
[0x00400030]    0x810b0001  lb $11, 1($8)                ; 12: lb    $t3, 1($t0)
[0x00400034]    0x810c0002  lb $12, 2($8)                ; 13: lb    $t4, 2($t0)

        DATA
[0x10000000]...[0x10010000]   00000000
[0x10010000]                  0x64636261
[0x10010010]                  0x79787675
[0x10010020]...[0x10040000]   0x00000000

        STACK
[0x7fffeffc]                  0x00000000

[0x0040000c]    0x00041080  sll $2, $4,
[0x00400010]    0x00c23021  addu $6, $6,
[0x00400014]    0x0c100009  jal 0x004000
[0x00400024]    0x3c081001  lui $8, 4097
[0x00400028]    0x8d090000  lw $9, 0($8)
[0x0040002c]    0x810a0000  lb $10, 0($8
[0x00400030]    0x810b0001  lb $11, 1($8
```

$t3 is 0x00000062

Load Byte

The content at address 0x1001000 is 0x64636261

Single step again...

PCSpim

File Simulator Window Help

```
PC       = 00400038   EPC      = 00000000   Cause    = 00000000   BadVAddr= 00000000
Status   = 3000ff10   HI       = 00000000   LO       = 00000000
                                General Registers
R0  (r0) = 00000000  R8  (t0) = 10010000  R16 (s0) = 00000000
R1  (at) = 00000000  R9  (t1) = 64636261  R17 (s1) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000061  R18 (s2) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000063  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000063  R20 (s4) = 00000000  R28 (gp) = 10008000
```

$t4 is 0x00000063

```
[0x0040001c]    0x3402000a  ori $2, $0, 10          ; 183: li $v0 10
[0x00400020]    0x0000000c  syscall                 ; 184: syscall                # syscall
[0x00400024]    0x3c081001  lui $8, 4097 [STR]      ; 8: la     $t0, STR
[0x00400028]    0x8d090000  lw $9, 0($8)            ; 9: lw     $t1, 0($t0)
[0x0040002c]    0x810a0000  lb $10, 0($8)           ; 11: lb    $t2, 0($t0)
[0x00400030]    0x810b0001  lb $11, 1($8)           ; 12: lb    $t3, 1($t0)
[0x00400034]    0x810c0002  lb $12, 2($8)           ; 13: lb    $t4, 2($t0)
[0x00400038]    0x810d0003  lb $13, 3($8)           ; 14: lb    $t5, 3($t0)
```

Load Byte

```
[0x10000000]...[0x10010000]
[0x10010000]                0x64636261
[0x10010010]                0x79787675
[0x10010020]...[0x10040000]  0x00000000

        STACK
[0x7fffeffc]                0x00000000
```

The content at address 0x1001000 is 0x64636261

```
[0x00400010]    0x00c23021  addu $6, $6,
[0x00400014]    0x0c100009  jal 0x0040002
[0x00400024]    0x3c081001  lui $8, 4097
[0x00400028]    0x8d090000  lw $9, 0($8)
[0x0040002c]    0x810a0000  lb $10, 0($8)
[0x00400030]    0x810b0001  lb $11, 1($8)
[0x00400034]    0x810c0002  lb $12, 2($8)
```

Single step again...

PCSpim

File Simulator Window Help

```
                                  General Registers
R0  (r0) = 00000000  R8  (t0) = 10010000  R16 (s0) = 00000000  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 64636261  R17 (s1) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000061  R18 (s2) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000062  R19 (s3) = 00000000
R4  (a0) = 00000000                       R20 (s4) = 00000000
R5  (a1) = 7ffff000  R13 (t5) = 00000064  R21 (s5) = 00000000  R29 (sp) = 7fffeffc
R6  (a2) = 7ffff004                       R22 (s6) = 00000000  R30 (s8) = 00000000
```

$t5 is 0x00000064

```
[0x00400020]    0x0000000c  syscall                          ; 184: syscall                syscall
[0x00400024]    0x3c081001  lui $8, 4097 [STR]               ; 8: la     $t0, STR
[0x00400028]    0x8d090000  lw $9, 0($8)                     ; 9: lw     $t1, 0($t0)
[0x0040002c]    0x810a0000  lb $10, 0($8)                    ; 11: lb    $t2, 0($t0)
[0x00400030]    0x810b0001  lb $11, 1($8)                    ; 12: lb    $t3, 1($t0)
[0x00400034]    0x810c0002  lb $12, 2($8)                    ; 13: lb    $t4, 2($t0)
[0x00400038]    0x810d0003  lb $13, 3($8)                    ; 14: lb    $t5, 3($t0)
[0x0                    08  jr $31                           ; 16: jr    $ra
```

Load Byte

```
        DATA
[0x10000000]...[0x10010000]
[0x10010000]                0x64636261
[0x10010010]                            0x79787675
[0x10010020]...[0x10040000]  0x00000000

        STACK
[0x7fffeffc]                0x000000
```

The content at address 0x1001000 is 0x64636261

```
[0x00400014]    0x0c100009  jal 0x004000                     jal main
[0x00400024]    0x3c081001  lui $8, 4097                     la
[0x00400028]    0x8d090000  lw $9, 0($8)                     lw
[0x0040002c]    0x810a0000  lb $10, 0($8                     lb
[0x00400030]    0x810b0001  lb $11, 1($8                     lb
[0x00400034]    0x810c0002  lb $12, 2($8                     lb
[0x00400038]    0x810d0003  lb $13, 3($8                     lb
```

Single step again...

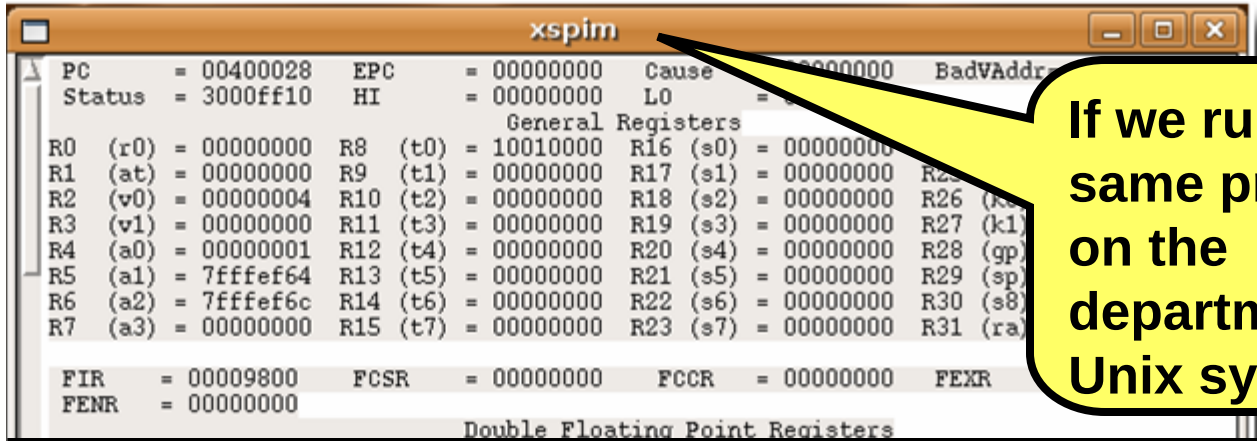
PCSpim

File Simulator Window Help

```
                                    General Registers
R0  (r0) = 00000000  R8  (t8) = 10010000  R16 (s0) = 00000000  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 64636261  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000061  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000062  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000063  R20 (s4) = 00000000  R28 (gp) = 10008000
R5  (a1) = 7ffff000  R13 (t5) = 00000064  R21 (s5) = 00000000  R29 (sp) = 7fffeffc
R6  (a2) = 7ffff004  R14 (t6) = 00000000  R22 (s6) = 00000000  R30 (s8) = 00000000

[0x00400020]    0x0000000c  syscall                          ; 184: syscall                    # syscall
[0x00400024]    0x3c081001  lui $8, 4097 [STR]               ; 8: la     $t0, STR
[0x00400028]    0x8d090000  lw $9, 0($8)                     ; 9: lw     $t1, 0($t0)
[0x0040002c]    0x810a0000  lb $10, 0($8)                    ; 11: lb    $t2, 0($t0)
[0x00400030]    0x810b0001
[0x00400034]    0x810c0002
[0x00400038]    0x810d0003
[0x0040003c]    0x02c00008

        DAT
[0x10000000
[0x10010000
[0x10010010
[0x10010020

        STA
[0x7fffeffc]                0x00000000

[0x00400014]    0x0c100009  jal 0x00400024 [main]            ; 180: jal main
[0x00400024]    0x3c081001  lui $8, 4097 [STR]               ; 8: la     $t0, STR
[0x00400028]    0x8d090000  lw $9, 0($8)                     ; 9: lw     $t1, 0($t0)
[0x0040002c]    0x810a0000  lb $10, 0($8)                    ; 11: lb    $t2, 0($t0)
[0x00400030]    0x810b0001  lb $11, 1($8)                    ; 12: lb    $t3, 1($t0)
[0x00400034]    0x810c0002  lb $12, 2($8)                    ; 13: lb    $t4, 2($t0)
[0x00400038]    0x810d0003  lb $13, 3($8)                    ; 14: lb    $t5, 3($t0)
```

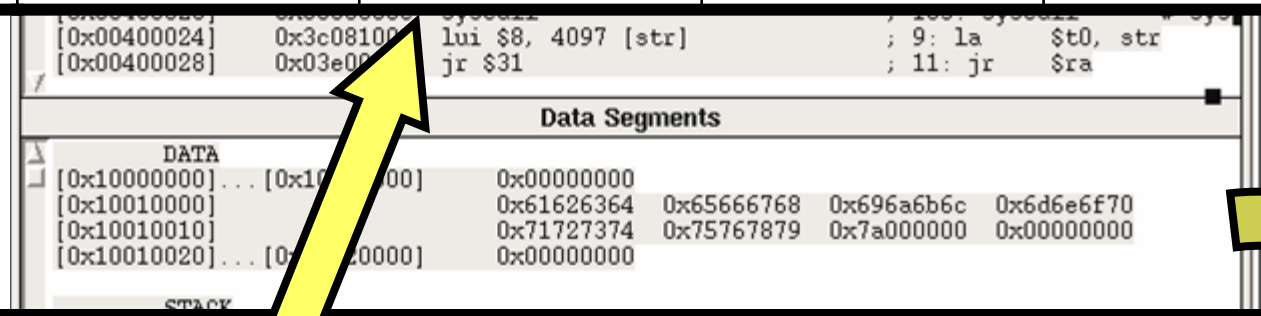
Word stored in $t1

Character	'd'	'c'	'b'	'a'
ASCII	0x64	0x63	0x62	0x61
Address	0x10010003	0x10010002	0x10010001	0x10010000



If we run the same program on the department Unix system

Word stored in $t1			
Character	'a'	'b'	'c'	'd'
ASCII	0x61	0x62	0x63	0x64
Address	0x10010000	0x10010001	0x10010002	0x10010003



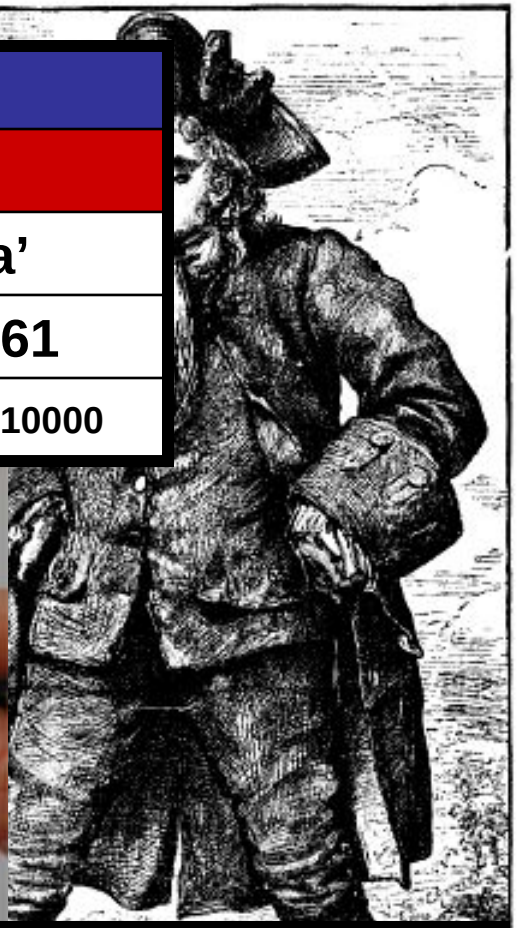

```
[0x10010000]  0x61626364 0x65666768 0x696a6b6c 0x6d6e6f70
[0x10010010]  0x71727374 0x75767778 0x797a0000 0x00000000
```

x68			
1 Word			
Character	’d’	’c’	’b’	’a’
ASCII	0x64	0x63	0x62	0x61
Address	0x10010003	0x10010002	0x10010001	0x10010000

What? We store the same string and gets two different words?

SPARC			
1 Word			
Character	’a’	’b’	’c’	’d’
ASCII	0x61	0x62	0x63	0x64
Address	0x10010000	0x10010001	0x10010002	0x10010003

x68 (Little Endian)				
1 Word				
Character	'd'	'c'	'b'	'a'
ASCII	0x64	0x63	0x62	0x61
Address	0x10010003	0x10010002	0x10010001	0x10010000



SPARC (Big Endian)				
1 Word				
Character	'a'	'b'	'c'	'd'
ASCII	0x61	0x62	0x63	0x64
Address	0x10010000	0x10010001	0x10010002	0x10010003

A MIPS processor can be configured to use either Litte Endian or Big Endian byte order.

SPIM is a simulator and uses the same byte order as the host machine.

print_string_and_integer.s

SPIM comes with a number of *system calls*.

```
        .data
STR:    .asciiz "Hello world, your lucky number is: "
        .text
        .globl main
main:
        # system call code for print_str
        li      $v0, 4
        # address of string to print
        la      $a0, STR
        syscall

        # system call code for print_int
        li      $v0, 1
        # integer to print
        addi    $a0, $zero, 44
        syscall

        jr      $ra
```

Print String

call code 4 in $v0

Address to string

Print number

call code 1 in $v0

Number to print

PCSpim

File Simulator Window Help

```
PC       = 00000000   EPC      = 00000000   Cause    = 00000000   BadVAddr= 00000000
Status   = 30006618   HI       = 00000000   LO       = 00000000

R0  (r                                      Registers
R1  (a                                      16 (s0)
R2  (v                                      17 (s1)
R3  (v                                      18 (s2)
R4  (a                                      19 (s3)

[0x00400000]     0x27a50004  addiu $5, $29, 4
[0x00400004]     0x24a60004  addiu $6, $5, 4
[0x00400008]     0x00041080  sll $2, $4, 2
[0x0040000c]     0x00c23021  addu $6, $6, $2
[0x00400010]     0x0c100009  jal 0x00400024 [main]
[0x00400014]     0x00000000  nop
[0x00400018]     0x3402000a  ori $2, $0, 10
[0x0040001c]

        DATA
[0x10000000]...[0x10010000]      0x00000000
[0x10010000]                     0x6c6c6548   0x6f772
[0x10010010]                     0x756c2072   0x20796
[0x10010020]                     0x00203a73   0x00000
[0x10010030]...[0x10040000]      0x00000000

        STACK

Copyright 1990-2004 by James R. Larus (larus@cs.wis
All Rights Reserved.
DOS and Windows ports by David A. Carley (dac@cs.wi
Copyright 1997 by Morgan Kaufmann Publishers, Inc.
See the file README for a full copyright notice.
Loaded: C:\Program Files\PCSpim\exceptions.s
C:\Documents and Settings\Karl Marklund\My Documents\Teaching\Digitalteknik och Datorarkitektur vt 2008 (1DT033
```

When you run this program, a console will pop up.

Console

```
Hello world, your lucky number is: 44
```

Both the string and the number is written to the console.

For Help, press F1 PC=0x00000000 EPC=0x00000000 Cause=0x00000000

start 6 Wi... 12 M... 7 Fir... untitle... OmTe... Calcul... 2 pc... 2 G... EN 4:35 AM


We often wants to repeat something a number of times...
It’s time to learn how to program loops.

loop.s

```
        .text
        .globl main

main:   add     $t0, $zero, $zero        # loop counter i
        addi    $s0 ,$zero, 5            # loop limit N

loop:   beq $t0, $s0, done               # for i = 0...(N-1)

        # integer to print
        add     $a0, $zero, $t0
        # system call code for print_int
        li      $v0, 1
        syscall

        addi $t0, $t0, 1                 # i ++

        j loop

done:   jr      $ra
```

Example of good comments

Branch on EQual: *beq*

If $t0 == $t5, jump to the label done.

Otherweise, continue with next instruction below.

Jump (j)

Unconditional jump – jump to the label loop.

loop.s

```
        .text
        .globl main

main:                   # loop counter i
                        # loop limit N

loop:                   # for i = 0...(N-1)

                        # print i
                        nt

                        # i ++

                        #  loop again

done:                   #  return to caller
```

Example of good comments

Introduce *abstractions* in your comments.

Refer to previously defined. *abstractions*.

Choose descriptive names for your labels.

You should be able to understand what the program does by only looking at the comments and the labels.

loop.s

```
        .text
        .globl main

main:                   # loop-counter i
                        # loop limit N

loop:                   # for i = 0...(N-1)


                        # print i
                        nt


                        # i ++

                        #  loop again

done:                   #  return to caller
```

If you start with the comments and the labels you can use them as your ”recipe” for the program, translating the comments and labels into MIPS assembly

loop.s

```
        .text
        .globl main

main:   add     $t0, $zero, $zero        # loop-counter i
        addi    $s0 ,$zero, 5            # loop limit N

loop:   beq $t0, $s0, done               # for i = 0...(N-1)


        add     $a0, $zero, $t0           # print i
        # system call code for print_int
        li      $v0, 1
        syscall

        addi $t0, $t0, 1                 # i ++

        j loop                           #  loop again

done:   jr      $ra                      #  return to caller
```

PCSpim

File Simulator Window Help

Load the source file loop.s

```
PC      = 00400000   [illegible] = 00000000   Cause    = 00000000   BadVAddr= 00000000
Status  = 3000ff10   HI          [illegible]  LO       = 00000000

R0  (r0) = 00000000  R8  (t[illegible]  [illegible]000000
R1  (at) = 00000000  R9  (t[illegible]  [illegible]000000
R2  (v0) = 00000000  R10 (t[illegible]  [illegible]000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000
```

```
[0x00400000]    0x8fa40000  lw $4, 0($29)              ; 175: lw $a0 0($sp)          # argc
[0x00400004]    0x27a50004  addiu $5, $29, 4           ; 176: addiu $a1 $sp 4        # argv
[0x00400008]    0x24a60004  addiu $6, $5, 4            ; 177: addiu $a2 $a1 4        # envp
[0x0040000c]    0x00041080  sll $2, $4, 2              ; 178: sll $v0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2            ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]      ; 180: jal main
[0x00400018]    0x00000000  nop                        ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10             ; 183: li $v0 10
```

```
        DATA
[0x10000000]...[0x10040000]     0x00000000

        STACK
[0x7fffeffc]                    0x00000000

        KERNEL DATA
[0x90000000]                    0x78452020  0x74706563  0x206e6f69  0x636f2000
```

```
Copyright 1990-2004 by James R. Larus (larus@cs.wisc.edu).
All Rights Reserved.
DOS and Windows ports by David A. Carley (dac@cs.wisc.edu).
Copyright 1997 by Morgan Kaufmann Publishers, Inc.
See the file README for a full copyright notice.
Loaded: C:\Program Files\PCSpim\exceptions.s
C:\Documents and Settings\Karl Marklund\My Documents\Teaching\Dark ht 2008\Tutorials By Karl Marklund\Part
```

PCSpim

File Simulator Window Help

```
PC     = 0040002c   EPC      = 00000000   Cause    = 00000000   BadVAddr= 00000000
Status = 3000ff10   HI       = 00000000   LO       = 00000000
                              General Registers
R0  (r0) = 00000000  R8  (t0) = 00000000  R16 (s0) = 00000005  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 00000000  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000000  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000

[0x00400010]    0x00c23021  addu $6, $6, $2                  ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]            ; 180: jal main
[0x00400018]    0x00000000  nop                              ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10                   ; 183: li $v0 10
[0x00400020]    0x0000000c  syscall                          ; 184: syscall                    # syscall 10 (ex
[0x00400024]    0x00004020  add $8, $0, $0                   ; 4: add    $t0, $zero, $zero     # loop-counter :
[0x00400028]    0x20100005  addi $16, $0, 5                  ; 5: addi   $s0 ,$zero, 5         # loop limit N
[0x0040002c]    0x11100006  beq $8, $16, 24 [done-0x0040002c]; 7: beq $t0, $s0, done    # for i = 0...4 = 0...(b

        DATA
[0x10000000]...[0x10040000]          00000000

        STACK
[0x7fffeffc]                         000

        KERNEL DATA
[0x90000000]                                                 0x636f2000

[0x00400004]    0x27a50                                      addiu $a1 $sp 4                   # argv
[0x00400008]    0x24a60004  addiu $6, $5, 4                  ; 177: addiu $a2 $a1 4            # envp
[0x0040000c]    0x00041080  sll $2, $4, 2                    ; 178: sll $v0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2                  ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]            ; 180: jal main
[0x00400024]    0x00004020  add $8, $0, $0                   ; 4: add    $t0, $zero, $zero     # loop-counter :
[0x00400028]    0x20100005  addi $16, $0, 5                  ; 5: addi   $s0 ,$zero, 5         # loop limit N
```

Single step until you reach this instruction.

For Help, press F1 PC=0x0040002c EPC=0x00000000 Cause=0x00000000

start 6 Wi... 12 M... 7 Fir... untitle... OmTe... Calcul... 2 pc... 2 G... SV 4:54 AM

PCSpim

File

Machine Instruction: 0x11100006

```
PC      = 00                    = 00000000   Cause    = 00000000   BadVAddr= 00000000
Status  = 300                   = 00000000   I0       = 00000000
                                  Genera
R0  (r0) = 000                  (0) = 0000000
R1  (at) = 0000                 (t1) = 0000000
R2  (v0) = 0000                 (t2) = 0000000
R3  (v1) = 00000                (t3) = 0000000
R4  (a0) = 00000                (t4) = 0000000

[0x00400010]    0                021  addu $6, $6
[0x00400014]    0x               0009  jal 0x00400
[0x00400018]    0x               0000  nop
[0x0040001c]    0x               2000a  ori $2, $0,
[0x00400020]    0x               0000c  syscall                                    (ex
[0x00400024]    0x               004020  add $8, $0,
[0x00400028]    0x20100005  addi $16, $0, 5                ; 5: addi   $s0 ,$zero, 5          # loop limit N
[0x0040002c]    0x11100006  beq $8, $16, 24 [done-0x0040002c]; 7: beq $t0, $s0, done    # for i = 0...4 = 0...(
```

Hexadecimal	Decimal	Binary
0x11	$1*16^1 + 1 = 17$	0001 0001
0x10	$1*16^1 = 16$	0001 0000
0x00	0	0000 0000
0x06	$6*16^0 = 6$	0000 0110

Register 8

Register 16 .

Branch offset 6

000100 01000 10000 0000 0000 0000 0110

op

OP(beq) = 4

beq is an *I-type instruction*.

For Help, press F1

PC=0x0040002c EPC=0x00000000 Cause=0x00000000

start 6 Wi... 12 M... 7 Fir... untitle... OmTe... Calcul... 2 pc... 2 G... SV 4:54 AM

PCSpim

File

Machine Instruction: 0x11100006

```
PC      = 00                    = 00000000   Cause    = 00000000   BadVAddr= 00000000
Status  = 300                   = 00000000   Genera
R0  (r0) = 000                  (0) = 0000000
R1  (at) = 0000                 (t1) = 0000000
R2  (v0) = 0000                 (t2) = 0000000
R3  (v1) = 00000                (t3) = 0000000
R4  (a0) = 00000                (t4) = 0000000

[0x00400010]    0                021  addu $6, $6
[0x00400014]    0x               0009  jal 0x00400
[0x00400018]    0x               0000  nop
[0x0040001c]    0x               2000a  ori $2, $0,
[0x00400020]    0x               0000c  syscall
[0x00400024]    0x               004020  add $8, $0,
[0x00400028]    0x20100005  addi $16, $0, 5                ; 5: addi   $s0 ,$zero, 5          # loop limit N
[0x0040002c]    0x11100006  beq $8, $16, 24 [done-0x0040002c]; 7: beq $t0, $s0, done    # for i = 0...4 = 0...(
```

Hexadecimal	Decimal	Binary
0x11	$1*16^1 + 1 = 17$	0001 0001
0x10	$1*16^1 = 16$	0001 0000
0x00	0	0000 0000
0x06	$6*16^0 = 6$	0000 0110

Register 8

Register 16 .

Branch offset 6

000100 01000 10000 0000 0000 0000 0110

op	rs	rt	immediate

OP(beq) = 4

beq is an *I-type instruction*.

For Help, press F1

PC=0x0040002c EPC=0x00000000 Cause=0x00000000

start 6 Wi... 12 M... 7 Fir... untitle... OmTe... Calcul... 2 pc... 2 G... SV 4:54 AM

PCSpim

File Simulator Window Help

```
PC      = 00000000   EPC     = 00000000   Cause   = 00000000   BadVAddr= 00000000
Status  = 20006618   HI      = 00000000   LO      = 00000000

R0  (r                                    isters
R1  (a                                    6 (s0)
R2  (v                                    7 (s1)
R3  (v                                    8 (s2)
R4  (a                                    9 (s3)

[0x00400000]    0x27a50004  addiu $5, $29, 4
[0x00400004]    0x24a60004  addiu $6, $5, 4
[0x00400008]    0x00041080  sll $2, $4, 2
[0x0040000c]    0x00c23021  addu $6, $6, $2
[0x00400010]    0x0c100009  jal 0x00400024 [main]
[0x00400014]    0x00000000  nop
[0x00400018]    0x3402000a  ori $2, $0, 10
[0x0040001c]

        DATA
[0x10000000]...[0x10040000]     0x00000000

        STACK
[0x7fffef78]                    0x00000000  0x00000
[0x7fffef80]                    0x7fffefc9  0x7fffe
[0x7fffef90]                    0x7fffef09  0x7fffe
[0x7fffefa0]                    0x7fffeeb1  0x7fffe

Copyright 1990-2004 by James R. Larus (larus@cs.wis
All Rights Reserved.
DOS and Windows ports by David A. Carley (dac@cs.wi
Copyright 1997 by Morgan Kaufmann Publishers, Inc.
See the file README for a full copyright notice.
Loaded: C:\Program Files\PCSpim\exceptions.s
C:\Documents and Settings\Karl Marklund\My Documents\Teaching\Digitalteknik och Datorarkitektur vt 2008 (1DT033
```

When you run this program, a console will pop up.

Console

```
01234
```

The loop prints out the number 0,1,2,3 and 4

For Help, press F1 PC=0x00000000 EPC=0x00000000 Cause=0x00000000

start 6 Wi... 12 M... 7 Fir... untitle... OmTe... Calcul... 2 pc... 2 G... SV 4:46 AM


To make choices we can use an IF-THEN-ELSE construct

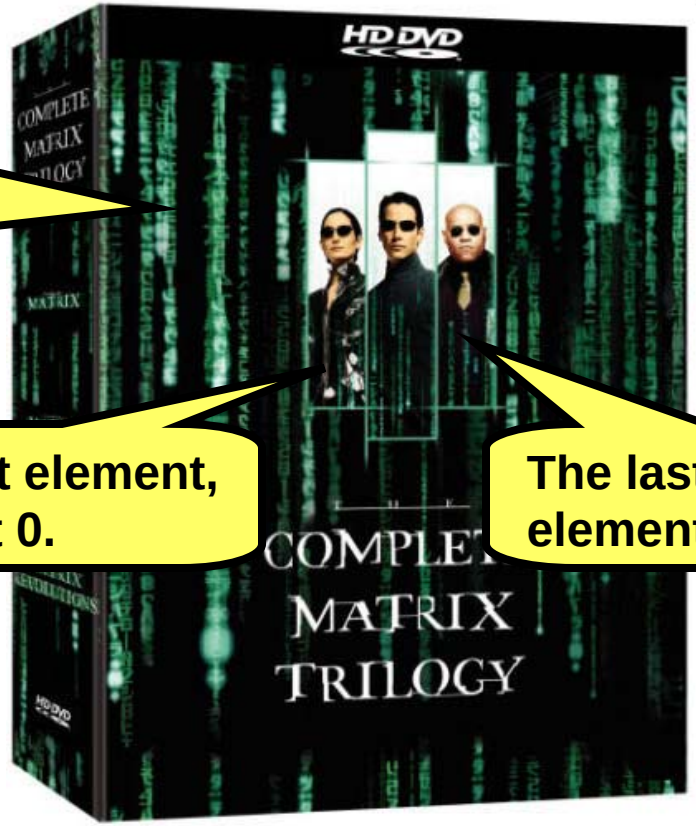
if_then_else.s

Branch Not Equal: *bne*

Load this program into SPIM and experiment wiht different values for a & b. Single step to follow the execution.

```
        .data
STR_THEN:    .asciiz "equal"
STR_ELSE:    .asciiz "not equal"
        .text
        .globl main

main:   li      $t0, 15                 # a
        addi    $t1, $zero, 15          # b

if:     bne $t0, $t1, else              # if (a==b)

then:   # system call for print_str     # print equal

        li      $v0, 4
        la      $a0, STR_THEN
        syscall
        j       end_if
else:
        # system call for print_str     # print not equal
        li      $v0, 4
        la      $a0, STR_ELSE
        syscall

end_if: jr      $ra
```

We often need to store things in a sequence



The first element, element 0.

The last element, element N-1

A sequence of N elements - an array.

array.s

```
        .data

ARRAY:  .word 1, 1, 2, 3, 5, 8, 13, 21, 34, 55
```

ARRAY is a label, i.e, the address to the first number.

10 numbers stored in sequence in the data segments – an array of numbers.

Each number is a word – i.e, four bytes

array.s

ARRAY:	.word	1	1	2	3	5	8	13	21	34	55
Index i		0	1	2	3	4	5	6	7	8	9

Let i be an index such that
the first number has index 0,
the second number has index 1,
the third number has index 2, ...

array.s

Address = ARRAY + 4

ARRAY:	.word	1	1	2	3	5	8	13	21	34	55
(index i)		0	1	2	3	4	5	6	7	8	9

Address = ARRAY

The label ARRAY is a named address in the data segment.

Address = ARRAY + 4 + 4 = ARRAY + 8

array.s

ARRAY:	.word	1	1	2	3	5	8	13	21	34	55
(index i)		0	1	2	3	4	5	6	7	8	9

For the ith element ARRAY[i], the address is ARRAY + 4*i

A clever way of multiplying by 2...

Multiplying by 2 is equivalent to shift 1 bit to the left.

Multiplying by 4 is equivalent to shift 2 bits to the left.

Example:

	5_{10}	=	00101_2	
$2*5_{10}=$	10_{10}	=	01010_2	Same as 00101 << 1 (shift left 1 bit)
$2*2*5_{10} = 4*5_{10}=$	20_{10}	=	10100_2	Same as 01010 << 1 (shift left 1 bit) Same as 00101 << 2 (shift left 2 bits)

array.s

```
        .data
NL:     .asciiz "\n"
SIZE:   .word 10
ARRAY:  .word 1, 1, 2, 3, 5, 8, 13, 21, 34, 55
        .text
        .globl main
main:   add     $t0, $zero, $zero   # array index i
        la      $t1, ARRAY
        lw      $t2, SIZE
loop:   beq     $t0, $t2, done      # for e in ARRAY
        sll     $t4, $t0, 2         # offset = 4*i
        add     $t3, $t1, $t4       # addr = ARRAY + offset
        lw      $a0, 0($t3)         # e =  ARRAY[i]
        li      $v0, 1              # print e
        syscall
        li      $v0, 4              # print  \n
        la      $a0, NL
        syscall

        addi    $t0, $t0, 1         # next element

        j       loop

done:   jr      $ra
```

Shif Left Logical: *sll*

Multiply by 4 is equivalent to shift left 2 bits.

PCSpim

File Simulator Window Help

Console

```
1
1
2
3
5
8
13
21
34
55
```

```
R0  (r0) = 00000000  R8  (t0)
R1  (at) = 10010000  R9  (t1)
R2  (v0) = 0000000a  R10 (t2)
R3  (v1) = 00000000  R11 (t3)
R4  (a0) = 10010000  R12 (t4)
R5  (a1) = 7ffff000  R13 (t5)
R6  (a2) = 7ffff004  R14 (t6)
```

```
(t8) = 00000000
(t9) = 00000000
(k0) = 00000000
(k1) = 00000000
(gp) = 10008000
(sp) = 7fffeffc
(s8) = 00000000
```

```
[0x00400018]    0x00000000  n
[0x00
[0x0
[0x0
[0x0
[0x0
[0x0
```

```
nop
li $v0 10
syscall                          # syscall
d    $t0, $zero, $zero           # array inc
     $t1, ARRAY
 [ARRAY]
 8 [ARRAY]
7
1)
```

When you run this program, a console will pop up.

All the numbers in the array are printed. A new line separate each number.

```
        DATA
[0x10000000]...[0x10010000]     0x00000000
[0x10010000]                    0x0000000a  0x0000000a
[0x10010010]                    0x00000002  0x00000003
[0x10010020]                    0x0000000d  0x00000015
[0x10010030]...[0x10040000]     0x00000000

        STACK
```

```
[0x0040005c]    0x21080001  addi $8, $8, 1                    ; 20: addi  $t0, $t0, 1          # next eler
[0x00400060]    0x0810000e  j 0x00400038 [loop]               ; 22: j     loop
[0x00400038]    0x110a000b  beq $8, $10, 44 [done-0x00400038]; 10: beq  $t0, $t2, done        # for e in
[0x00400064]    0x03e00008  jr $31                            ; 24: jr    $ra
[0x00400018]    0x00000000  nop                               ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10                    ; 183: li $v0 10
[0x00400020]    0x0000000c  syscall                           ; 184: syscall                   # syscall
```

You learned about arrays and strings. But what about arrays of strings...

array_of_string.s

Declear three strings

Each label denotes the start address of a string

Use these addresses to construct an array of strings.

```
        .data

STR_1:  .asciiz "The sum of "
STR_2:  .asciiz " and "
STR_3:  .asciiz " is "


ARRAY_OF_STRINGS:
        # Each element is an address to a string.
        .word STR_1, STR_2, STR_3
```

PCSpim

File Simulator Window Help

```
PC       = 0040002c   EPC      = 00000000   Cause    = 00000000   BadVAddr= 00000000
Status   = 3000ff10   HI       = 00000000   LO       = 00000000

                          General Registers
R0  (r0) = 00000000  R8  (t0) = 10010018  R16 (s0) = 00000000  R24 (t8) = 00000000
R1  (at) = 10010000  R9  (t1) = 00000000  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000000  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000

[0x00400010]    0x00c23021  addu $6, $6, $2                  ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]            ; 180: jal main
[0x00400018]    0x00000000  nop                              ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10                   ; 183: li $v0 10
[0x00400020]    0x0000000c  syscall                          ; 184: syscall                    #
[0x00400024]    0x3c011001  lui $1, 4097 [ARRAY_OF_STRINGS]  ; 17: la    $t0, ARRAY_OF_STRINGS
[0x00400028]    0x34280018  ori $8, $1, 24 [ARRAY_OF_STRINGS]
[0x0040002c]    0x20050003  addi $5, $0, 3                   ; 18: addi  $a1, $zero, 3   # a

        DATA
[0x10000000]...[0x10010000]     0x00000000
[0x10010000]                    0x20656854  0x206d7573  0x00206661  0x6
[0x10010010]                    0x69200020  0x00002073  0x10010000  0x1
[0x10010020]                    0x10010012  0x00000000  0x00000000  0x00000
[0x10010030]...[0x10040000]     0x00000000

[0x00400004]    0x27a50004  addiu $5, $9, 4                                          #
[0x00400008]    0x24a60004  addiu $6, $5, 4                                          #
[0x0040000c]    0x
[0x00400010]    0x
[0x00400014]    0x0c100009  jal 0x00400024 [main]
[0x00400024]    0x3c011001  lui $1, 4097 [ARRAY_OF_STRINGS]  ; 17: la    $t0, ARRAY_OF_STRINGS
[0x00400028]    0x34280018  ori $8, $1, 24 [ARRAY_OF_STRINGS]
```

Address 0x10010010

Address 0x10010014

The label ARRAY_OF_STRINGS refers to address 0x10010018

On this address in memor, the address to the stirng STR_1 is stored.

array_of_string.s (text segment part one)

```
        .text
        .globl main
main:
        # Just for fun,  get  the address of
        # label "ARRAY_OF_STRINGS":
        la      $t0, ARRAY_OF_STRINGS

        addi    $a1, $zero, 3       # a
        addi    $a2, $zero, 11      # b

        # Must copy $a0 since the
        # syscalls used later needs $a0

        add     $t0, $a0, $zero

        # Print "The sum of "
        li      $v0, 4
        lw      $a0, ARRAY_OF_STRINGS
        syscall

        # Print the value of a
        li      $v0, 1
        add     $a0, $zero, $a1
        syscall

        # Print  " and "
        li      $v0, 4
        lw      $a0, ARRAY_OF_STRINGS + 4
        syscall
```

Get address to STR_1 ”The sum of ”

Get address to STR_2 ” and ”

Address to the 2nd element in ARRAY_OF_STIRINGS

array_of_string.s (text segment part two)

```
# Print the value of b
li          $v0, 1
add         $a0, $zero, $a2
syscall

# Print " is "
li      $v0, 4
lw          $a0, ARRAY_OF_STRINGS + 8
 syscall

# Print the sum a + b
li          $v0, 1
add         $a0, $a1, $a2
syscall

jr          $ra
```

Get address to STR_3 ” is ”

Address to the 3rd element in ARRAY_OF_STIRINGS

PCSpim

File Simulator Window Help

Console

The sum of

```
PC       = 00400040   EPC      = 00000000   Cause    = 00000000
Status   = 3000ff10   HI       = 00000000   LO       = 00000000
                          General Registers
R0  (r0) = 00000000   R8  (t0) = 00000000   R16 (s0) = 00000000   R24 (t8) = 00000000
R1  (at) = 10010000   R9  (t1) = 00000000   R25 (t9) = 00000000
R2  (v0) = 00000004   R10 (t2) = 00000000   R26 (k0) = 00000000
R3  (v1) = 00000000   R11 (t3) = 00000000   R27 (k1) = 00000000
R4  (a0) = 10010000   R12 (t4) = 00000000
```

```
[0x00400024]    0x20050003  addi $5, $0, 3                    ; 17: addi   $s0, $zero, 3   # a
[0x00400028]    0x2006000b  addi $6, $0, 11                   ; 18: addi   $s1, $zero, 11  # b
[0x0040002c]    0x00804020  add $8, $4, $0                    ; 21: add    $s2, $a0, $zero
[0x00400030]    0x34020004  ori $2, $0, 4                     ; 24: li     $v0, 4
[0x00400034]    0x3c011001  lui $1, 4097                      ; 25: lw     $a0, ARRAY_OF_STRINGS
[0x00400038]    0x[illegible]                                 ; 26: syscall
[0x0040003c]    [illegible]                                   ; 29: li     $v0, 1
[0x00400040]    [illegible] $0
```

```
DATA
[0x10000000]...[0x10010000]   0x00000000
[0x10010000]                  0x20656854   0x206d7573   0x0020666f   0x646e6120
[0x10010010]                  0x69200020   0x00002073   0x10010000   0x1001000c
[0x10010020]                  0x10010012   0x00000000   0x00000000   0x00000000
[0x10010030]...[0x10040000]   0x00000000
```

m u s f o

ASCII('T') = 0x54

ASCII('h') = 0x68

ASCII('e') = 0x65

The string is *terminated* by 0x00 (NUL).

The label ARRAY_OF_STRINGS refers to address 0x10010018


```
[0x00400024]    0x20050003  addi $5, $0, 3    ; 17: addi   $s0, $zero, 3   # a
[0x00400028]    0x2006000b  addi $6, $0, 11   ; 18: addi   $s1, $zero, 11  # b
[0x0040002c]    0x00804020  add $8, $4, $0    ; 21: add    $s2, $a0, $zero
[0x00400030]    0x34020004  ori $2, $0, 4     ; 24: li     $v0, 4
[0x00400034]    0x3c011001  lui $1, 4097      ; 25: lw     $a0, ARRAY_OF_STRINGS
```

PCSpim

File Simulator Window Help

Console

```
The sum of 3 and
```

```
PC       = 00400064   EPC      = 00000000   Cause    = 00000000
Status   = 3000ff10   HI       = 00000000   LO       = 00000000
                          General Registers
R0  (r0) = 00000000  R8  (t0) = 00000000  R16 (s0) = 00000000  R24 (t8) = 00000000
R1  (at) = 10010000  R9  (t1) = 00000000  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000004  R10 (t2) = 00000000  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 1001000c  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000
```

```
[0x00400048]    0x34020001  ori $2, $0, 1                  ; 30: li    $v0, 1
[0x0040004c]    0x00052020  add $4, $0, $5                  ; 31: add   $a0, $zero, $a1
[0x00400050]    0x0000000c  syscall                         ; 32: syscall
[0x00400054]    0x34020004  ori $2, $0, 4                   ; 35: li    $v0, 4
[0x00400058]    0x3c011001  lui $1, 4097                    ; 36: lw    $a0, ARRAY_OF_STRINGS + 4
[0x0040005c]    0x8c24001c  lw $4, 28($1)
[0x00400060]    0x0000000c  syscall                         ; 37: syscall
[0x00400064]    0x34020001  ori $2, $0, 1                   ; 40: li    $v0, 1
```

```
        DATA
[0x10000000]...[0x10010000]     0x00000000
[0x10010000]                    0x20656854  0x206d7573  0x0020666f  0x646e6120
```

```
        DATA
[0x10000000]...[0x10010000]     0x00000000
[0x10010000]                    0x20656854  0x206d7573  0x0020666f  0x646e6120
[0x10010010]                    0x69200020  0x00002073  0x10010000  0x1001000c
[0x10010020]                    0x10010012  0x00000000  0x00000000  0x00000000
[0x10010030]...[0x10040000]     0x00000000
```

d n a

ARRAY_OF_STRINGS + 4

```
[0x00400050]    0x0000000c  syscall                         ; 32
[0x00400054]    0x34020004  ori $2, $0, 4                   ; 35
[0x00400058]    0x3c011001  lui $1, 4097                    ; 36
[0x0040005c]    0x8c24001c  lw $4, 28($1)
[0x00400060]    0x0000000c  syscall                         ; 37: syscall
```


You are now done with part four of your MIPS assembly programming training.