

SPIM Tutorial

CSE 410 Computer Systems

Introduction

- **SPIM: MIPS simulator**
 - Reads/executes assembly source programs
- **Does not execute binaries**
- Download
 - <http://www.cs.wisc.edu/~larus/spim.html>
 - Windows: PCSpim
 - Linux: xspim
- Read
 - Hennessy & Patterson, Appendix A
 - Resources at SPIM web site

Environment

The screenshot shows the PCSpim simulator interface with the following sections:

- Registers:** Shows the General Registers (R0-R31) with their current values.
- Text segment:** Shows assembly code for the main program, including instructions like lw, addiu, sll, addu, jal, and syscall.
- Data segment:** Shows the initial value of the stack at 0x7ffffeffc.
- Spim messages:** Displays copyright and version information for SPIM.

```
PC      = 00000000  EPC     = 00000000  Cause    = 00000000  BadVAddr= 00000000
Status   = 3000ff10  HI      = 00000000  LO      = 00000000
General Registers
R0 (r0) = 00000000  R8 (t0) = 00000000  R16 (s0) = 00000000  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
R5 (a1) = 00000000  R13 (t5) = 00000000  R21 (s5) = 00000000  R29 (sp) = 7ffffeffc
[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] 0x0c000000 jal 0x00000000 [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 (exit)
[0x10000000]...[0x10040000] 0x00000000
DATA
STACK
[0x7ffffeffc] 0x00000000
KERNEL DATA
[0x90000000] 0x78452020 0x74706563 0x206e6f69 0x636f2000
[0x90000010] 0x72727563 0x61206465 0x6920646e 0x726f6e67
SPIM Version Version 7.3 of August 26, 2006
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
For Help, press F1
PC=0x00000000 EPC=0x00000000 Cause=0x00000000
```

Registers

Number	Mnemonic	Usage	Number	Mnemonic	Usage
\$0	zero	Permanently 0	\$24, \$25	\$t8, \$t9	Temporary
\$1	\$at	Assembler Temporary	\$26, \$27	\$k0, \$k1	Kernel
\$2, \$3	\$v0, \$v1	Value returned by a subroutine	\$28	\$gp	Global Pointer
\$4-\$7	\$a0-\$a3	Subroutine Arguments	\$29	\$sp	Stack Pointer
\$8-\$15	\$t0-\$t7	Temporary	\$30	\$fp	Frame Pointer
\$16-\$23	\$s0-\$s7	Saved registers	\$31	\$ra	Return Address

Let's try

.text

.globl main

main:

li	\$t0, 0x2	# \$t0 ← 0x2
li	\$t1, 0x3	# \$t1 ← 0x3
addu	\$t2, \$t0, \$t1	# \$t2 ← ADD(\$t0, \$t1)

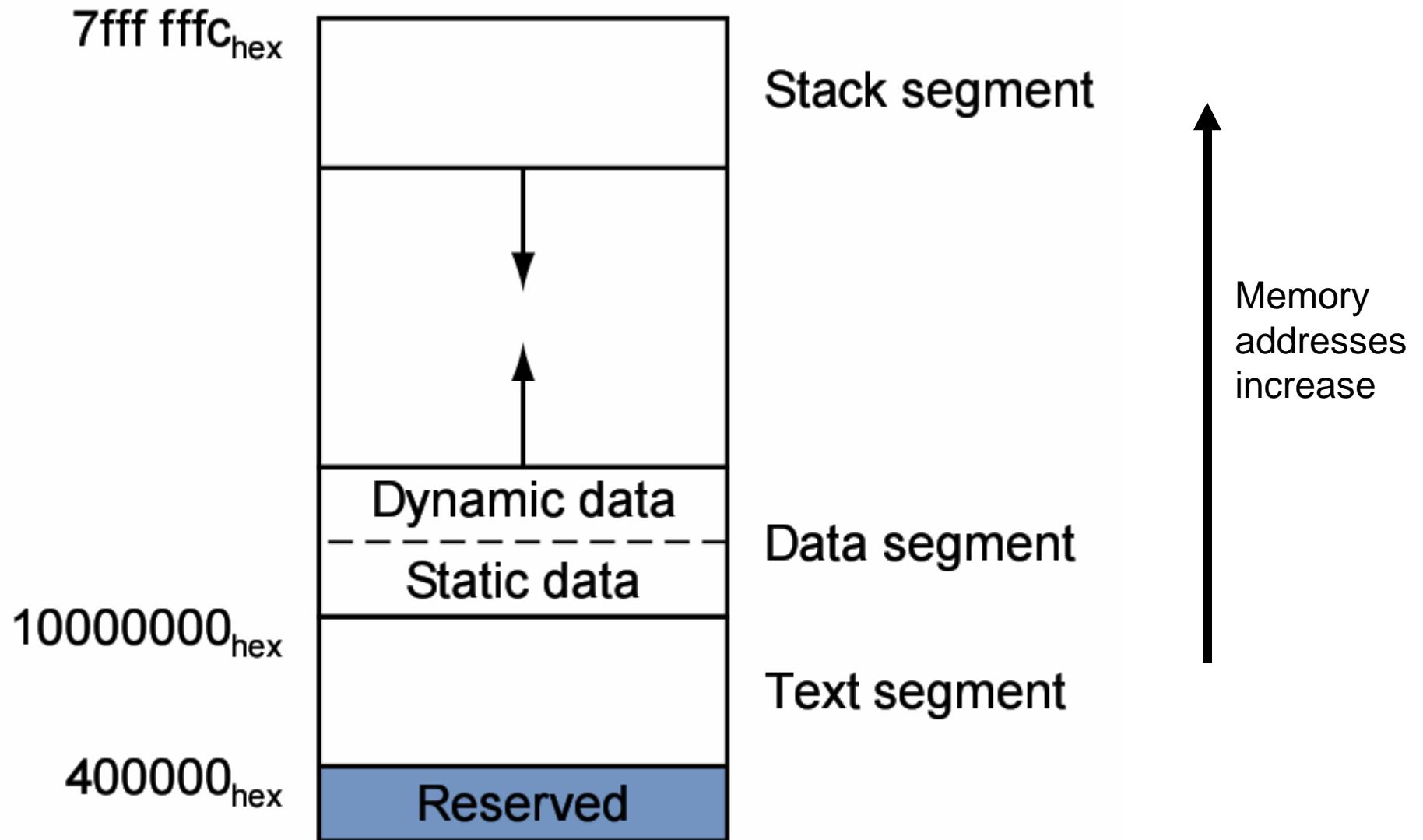
Let's try

```
.text  
.globl main
```

main:

ori	\$t0, \$0, 0x2	# \$t0 ← OR(0, 0x2)
ori	\$t1, \$0, 0x3	# \$t1 ← OR(0, 0x3)
addu	\$t2, \$t0, \$t1	# \$t2 ← ADD(\$t0, \$t1)

Memory layout



PC=0x00000000 EPC=0x00000000 Cause=0x00000000 Status=3000ff10 HI=00000000 LO=00000000 Registers

R0 (r0) = 00000000 R8 (t0) = 00000000 R16 (s0) = 00000000 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
R5 (a1) = 00000000 R13 (t5) = 00000000 R21 (s5) = 00000000 R29 (sp) = 7fffffe
R6 (v2) = 00000000 R14 (t6) = 00000000 R22 (s6) = 00000000 R30 (k2) = 00000000
R7 (v3) = 00000000 R15 (t7) = 00000000 R23 (s7) = 00000000 R31 (k3) = 00000000
R8 (v4) = 00000000 R16 (t8) = 00000000 R24 (t8) = 00000000 R32 (k4) = 00000000
R9 (v5) = 00000000 R17 (t9) = 00000000 R25 (t9) = 00000000 R33 (k5) = 00000000
R10 (v6) = 00000000 R18 (t10) = 00000000 R26 (k0) = 00000000 R34 (k6) = 00000000
R11 (v7) = 00000000 R19 (t11) = 00000000 R27 (k1) = 00000000 R35 (k7) = 00000000
R12 (v8) = 00000000 R20 (s2) = 00000000 R28 (gp) = 10008000 R36 (k8) = 00000000
R13 (v9) = 00000000 R21 (s3) = 00000000 R29 (sp) = 7fffffe R37 (k9) = 00000000
R14 (v10) = 00000000 R22 (s4) = 00000000 R30 (k2) = 00000000 R38 (k10) = 00000000
R15 (v11) = 00000000 R23 (s5) = 00000000 R31 (k3) = 00000000 R39 (k11) = 00000000
R16 (v12) = 00000000 R24 (t8) = 00000000 R32 (k4) = 00000000 R40 (k12) = 00000000
R17 (v13) = 00000000 R25 (t9) = 00000000 R33 (k5) = 00000000 R41 (k13) = 00000000
R18 (v14) = 00000000 R26 (k0) = 00000000 R34 (k6) = 00000000 R42 (k14) = 00000000
R19 (v15) = 00000000 R27 (k1) = 00000000 R35 (k7) = 00000000 R43 (k15) = 00000000
R20 (s4) = 00000000 R28 (gp) = 10008000 R36 (k8) = 00000000 R44 (k16) = 00000000
R21 (s5) = 00000000 R29 (sp) = 7fffffe R37 (k9) = 00000000 R45 (k17) = 00000000
R22 (s6) = 00000000 R30 (k2) = 00000000 R38 (k10) = 00000000 R46 (k18) = 00000000
R23 (s7) = 00000000 R31 (k3) = 00000000 R39 (k11) = 00000000 R47 (k19) = 00000000
R24 (t8) = 00000000 R32 (k4) = 00000000 R40 (k12) = 00000000 R48 (k20) = 00000000
R25 (t9) = 00000000 R33 (k5) = 00000000 R41 (k13) = 00000000 R49 (k21) = 00000000
R26 (k0) = 00000000 R34 (k6) = 00000000 R42 (k14) = 00000000 R50 (k22) = 00000000
R27 (k1) = 00000000 R35 (k7) = 00000000 R43 (k15) = 00000000 R51 (k23) = 00000000
R28 (gp) = 10008000 R36 (k8) = 00000000 R44 (k16) = 00000000 R52 (k24) = 00000000
R29 (sp) = 7fffffe R37 (k9) = 00000000 R45 (k17) = 00000000 R53 (k25) = 00000000
R30 (k2) = 00000000 R38 (k10) = 00000000 R46 (k18) = 00000000 R54 (k26) = 00000000
R31 (k3) = 00000000 R39 (k11) = 00000000 R47 (k19) = 00000000 R55 (k27) = 00000000
R32 (k4) = 00000000 R40 (k12) = 00000000 R48 (k20) = 00000000 R56 (k28) = 00000000
R33 (k5) = 00000000 R41 (k13) = 00000000 R49 (k21) = 00000000 R57 (k29) = 00000000
R34 (k6) = 00000000 R42 (k14) = 00000000 R50 (k22) = 00000000 R58 (k30) = 00000000
R35 (k7) = 00000000 R43 (k15) = 00000000 R51 (k23) = 00000000 R59 (k31) = 00000000
R36 (k8) = 00000000 R44 (k16) = 00000000 R52 (k24) = 00000000 R60 (k32) = 00000000
R37 (k9) = 00000000 R45 (k17) = 00000000 R53 (k25) = 00000000 R61 (k33) = 00000000
R38 (k10) = 00000000 R46 (k18) = 00000000 R54 (k26) = 00000000 R62 (k34) = 00000000
R39 (k11) = 00000000 R47 (k19) = 00000000 R55 (k27) = 00000000 R63 (k35) = 00000000
R40 (k12) = 00000000 R48 (k20) = 00000000 R56 (k28) = 00000000 R64 (k36) = 00000000
R41 (k13) = 00000000 R50 (k22) = 00000000 R57 (k29) = 00000000 R65 (k37) = 00000000
R42 (k14) = 00000000 R51 (k23) = 00000000 R58 (k30) = 00000000 R66 (k38) = 00000000
R43 (k15) = 00000000 R52 (k24) = 00000000 R59 (k31) = 00000000 R67 (k39) = 00000000
R44 (k16) = 00000000 R53 (k25) = 00000000 R60 (k32) = 00000000 R68 (k40) = 00000000
R45 (k17) = 00000000 R54 (k26) = 00000000 R61 (k33) = 00000000 R69 (k41) = 00000000
R46 (k18) = 00000000 R55 (k27) = 00000000 R62 (k34) = 00000000 R70 (k42) = 00000000
R47 (k19) = 00000000 R56 (k28) = 00000000 R63 (k35) = 00000000 R71 (k43) = 00000000
R48 (k20) = 00000000 R57 (k29) = 00000000 R64 (k36) = 00000000 R72 (k44) = 00000000
R49 (k21) = 00000000 R58 (k30) = 00000000 R65 (k37) = 00000000 R73 (k45) = 00000000
R50 (k22) = 00000000 R59 (k31) = 00000000 R66 (k38) = 00000000 R74 (k46) = 00000000
R51 (k23) = 00000000 R60 (k32) = 00000000 R67 (k39) = 00000000 R75 (k47) = 00000000
R52 (k24) = 00000000 R61 (k33) = 00000000 R68 (k40) = 00000000 R76 (k48) = 00000000
R53 (k25) = 00000000 R62 (k34) = 00000000 R69 (k41) = 00000000 R77 (k49) = 00000000
R54 (k26) = 00000000 R63 (k35) = 00000000 R70 (k42) = 00000000 R78 (k50) = 00000000
R55 (k27) = 00000000 R64 (k36) = 00000000 R71 (k43) = 00000000 R79 (k51) = 00000000
R56 (k28) = 00000000 R65 (k37) = 00000000 R72 (k44) = 00000000 R80 (k52) = 00000000
R57 (k29) = 00000000 R66 (k38) = 00000000 R73 (k45) = 00000000 R81 (k53) = 00000000
R58 (k30) = 00000000 R67 (k39) = 00000000 R74 (k46) = 00000000 R82 (k54) = 00000000
R59 (k31) = 00000000 R68 (k40) = 00000000 R75 (k47) = 00000000 R83 (k55) = 00000000
R60 (k32) = 00000000 R69 (k41) = 00000000 R76 (k48) = 00000000 R84 (k56) = 00000000
R61 (k33) = 00000000 R70 (k42) = 00000000 R77 (k49) = 00000000 R85 (k57) = 00000000
R62 (k34) = 00000000 R71 (k43) = 00000000 R78 (k50) = 00000000 R86 (k58) = 00000000
R63 (k35) = 00000000 R72 (k44) = 00000000 R79 (k51) = 00000000 R87 (k59) = 00000000
R64 (k36) = 00000000 R73 (k45) = 00000000 R80 (k52) = 00000000 R88 (k60) = 00000000
R65 (k37) = 00000000 R74 (k46) = 00000000 R81 (k53) = 00000000 R89 (k61) = 00000000
R66 (k38) = 00000000 R75 (k47) = 00000000 R82 (k54) = 00000000 R90 (k62) = 00000000
R67 (k39) = 00000000 R76 (k48) = 00000000 R83 (k55) = 00000000 R91 (k63) = 00000000
R68 (k40) = 00000000 R77 (k49) = 00000000 R84 (k56) = 00000000 R92 (k64) = 00000000
R69 (k41) = 00000000 R78 (k50) = 00000000 R85 (k57) = 00000000 R93 (k65) = 00000000
R70 (k42) = 00000000 R79 (k51) = 00000000 R86 (k58) = 00000000 R94 (k66) = 00000000
R71 (k43) = 00000000 R80 (k52) = 00000000 R87 (k59) = 00000000 R95 (k67) = 00000000
R72 (k44) = 00000000 R83 (k55) = 00000000 R88 (k60) = 00000000 R96 (k68) = 00000000
R73 (k45) = 00000000 R84 (k56) = 00000000 R89 (k61) = 00000000 R97 (k69) = 00000000
R74 (k46) = 00000000 R85 (k57) = 00000000 R90 (k62) = 00000000 R98 (k70) = 00000000
R75 (k47) = 00000000 R86 (k58) = 00000000 R91 (k63) = 00000000 R99 (k71) = 00000000
R76 (k48) = 00000000 R87 (k59) = 00000000 R92 (k64) = 00000000 R100 (k72) = 00000000
R77 (k49) = 00000000 R88 (k60) = 00000000 R93 (k65) = 00000000 R101 (k73) = 00000000
R78 (k50) = 00000000 R89 (k61) = 00000000 R94 (k66) = 00000000 R102 (k74) = 00000000
R79 (k51) = 00000000 R90 (k62) = 00000000 R95 (k67) = 00000000 R103 (k75) = 00000000
R80 (k52) = 00000000 R91 (k63) = 00000000 R96 (k68) = 00000000 R104 (k76) = 00000000
R81 (k53) = 00000000 R92 (k64) = 00000000 R97 (k69) = 00000000 R105 (k77) = 00000000
R82 (k54) = 00000000 R93 (k65) = 00000000 R98 (k70) = 00000000 R106 (k78) = 00000000
R83 (k55) = 00000000 R94 (k66) = 00000000 R99 (k71) = 00000000 R107 (k79) = 00000000
R84 (k56) = 00000000 R95 (k67) = 00000000 R100 (k72) = 00000000 R108 (k80) = 00000000
R85 (k57) = 00000000 R96 (k68) = 00000000 R101 (k73) = 00000000 R109 (k81) = 00000000
R86 (k58) = 00000000 R97 (k69) = 00000000 R102 (k74) = 00000000 R110 (k82) = 00000000
R87 (k59) = 00000000 R98 (k70) = 00000000 R103 (k75) = 00000000 R111 (k83) = 00000000
R88 (k60) = 00000000 R99 (k71) = 00000000 R104 (k76) = 00000000 R112 (k84) = 00000000
R89 (k61) = 00000000 R100 (k72) = 00000000 R105 (k77) = 00000000 R113 (k85) = 00000000
R90 (k62) = 00000000 R101 (k73) = 00000000 R106 (k78) = 00000000 R114 (k86) = 00000000
R91 (k63) = 00000000 R102 (k74) = 00000000 R107 (k79) = 00000000 R115 (k87) = 00000000
R92 (k64) = 00000000 R103 (k75) = 00000000 R108 (k80) = 00000000 R116 (k88) = 00000000
R93 (k65) = 00000000 R104 (k76) = 00000000 R109 (k81) = 00000000 R117 (k89) = 00000000
R94 (k66) = 00000000 R105 (k77) = 00000000 R110 (k82) = 00000000 R118 (k90) = 00000000
R95 (k67) = 00000000 R106 (k78) = 00000000 R111 (k83) = 00000000 R119 (k91) = 00000000
R96 (k68) = 00000000 R107 (k79) = 00000000 R112 (k84) = 00000000 R120 (k92) = 00000000
R97 (k69) = 00000000 R108 (k80) = 00000000 R113 (k85) = 00000000 R121 (k93) = 00000000
R98 (k70) = 00000000 R109 (k81) = 00000000 R114 (k86) = 00000000 R122 (k94) = 00000000
R99 (k71) = 00000000 R110 (k82) = 00000000 R115 (k87) = 00000000 R123 (k95) = 00000000
R100 (k72) = 00000000 R111 (k83) = 00000000 R116 (k88) = 00000000 R124 (k96) = 00000000
R101 (k73) = 00000000 R112 (k84) = 00000000 R117 (k89) = 00000000 R125 (k97) = 00000000
R102 (k74) = 00000000 R113 (k85) = 00000000 R118 (k90) = 00000000 R126 (k98) = 00000000
R103 (k75) = 00000000 R114 (k86) = 00000000 R119 (k91) = 00000000 R127 (k99) = 00000000
R104 (k76) = 00000000 R115 (k87) = 00000000 R120 (k92) = 00000000 R128 (k100) = 00000000
R105 (k77) = 00000000 R116 (k88) = 00000000 R121 (k93) = 00000000 R129 (k101) = 00000000
R106 (k78) = 00000000 R117 (k89) = 00000000 R122 (k94) = 00000000 R130 (k102) = 00000000
R107 (k79) = 00000000 R118 (k90) = 00000000 R123 (k95) = 00000000 R131 (k103) = 00000000
R108 (k80) = 00000000 R119 (k91) = 00000000 R124 (k96) = 00000000 R132 (k104) = 00000000
R109 (k81) = 00000000 R120 (k92) = 00000000 R125 (k97) = 00000000 R133 (k105) = 00000000
R110 (k82) = 00000000 R121 (k93) = 00000000 R126 (k98) = 00000000 R134 (k106) = 00000000
R111 (k83) = 00000000 R122 (k94) = 00000000 R127 (k99) = 00000000 R135 (k107) = 00000000
R112 (k84) = 00000000 R123 (k95) = 00000000 R128 (k96) = 00000000 R136 (k108) = 00000000
R113 (k85) = 00000000 R124 (k96) = 00000000 R129 (k97) = 00000000 R137 (k109) = 00000000
R114 (k86) = 00000000 R125 (k98) = 00000000 R130 (k100) = 00000000 R138 (k110) = 00000000
R115 (k87) = 00000000 R126 (k99) = 00000000 R131 (k101) = 00000000 R139 (k111) = 00000000
R116 (k88) = 00000000 R127 (k100) = 00000000 R132 (k102) = 00000000 R140 (k112) = 00000000
R117 (k89) = 00000000 R128 (k101) = 00000000 R133 (k103) = 00000000 R141 (k113) = 00000000
R118 (k90) = 00000000 R129 (k102) = 00000000 R134 (k104) = 00000000 R142 (k114) = 00000000
R119 (k91) = 00000000 R130 (k103) = 00000000 R135 (k105) = 00000000 R143 (k115) = 00000000
R120 (k92) = 00000000 R131 (k104) = 00000000 R136 (k106) = 00000000 R144 (k116) = 00000000
R121 (k93) = 00000000 R132 (k105) = 00000000 R137 (k107) = 00000000 R145 (k117) = 00000000
R122 (k94) = 00000000 R133 (k106) = 00000000 R138 (k108) = 00000000 R146 (k118) = 00000000
R123 (k95) = 00000000 R134 (k107) = 00000000 R139 (k109) = 00000000 R147 (k119) = 00000000
R124 (k96) = 00000000 R135 (k108) = 00000000 R140 (k110) = 00000000 R148 (k120) = 00000000
R125 (k97) = 00000000 R136 (k109) = 00000000 R141 (k111) = 00000000 R149 (k121) = 00000000
R126 (k98) = 00000000 R137 (k110) = 00000000 R142 (k112) = 00000000 R150 (k122) = 00000000
R127 (k99) = 00000000 R138 (k111) = 00000000 R143 (k113) = 00000000 R151 (k123) = 00000000
R128 (k100) = 00000000 R139 (k112) = 00000000 R144 (k114) = 00000000 R152 (k124) = 00000000
R129 (k101) = 00000000 R140 (k113) = 00000000 R145 (k115) = 00000000 R153 (k125) = 00000000
R130 (k102) = 00000000 R141 (k114) = 00000000 R146 (k116) = 00000000 R154 (k126) = 00000000
R131 (k103) = 00000000 R142 (k115) = 00000000 R147 (k117) = 00000000 R155 (k127) = 00000000
R132 (k104) = 00000000 R143 (k116) = 00000000 R148 (k118) = 00000000 R156 (k128) = 00000000
R133 (k105) = 00000000 R144 (k117) = 00000000 R149 (k119) = 00000000 R157 (k129) = 00000000
R134 (k106) = 00000000 R145 (k118) = 00000000 R150 (k120) = 00000000 R158 (k130) = 00000000
R135 (k107) = 00000000 R146 (k119) = 00000000 R151 (k121) = 00000000 R159 (k131) = 00000000
R136 (k108) = 00000000 R147 (k120) = 00000000 R152 (k122) = 00000000 R160 (k132) = 00000000
R137 (k109) = 00000000 R148 (k121) = 00000000 R153 (k123) = 00000000 R161 (k133) = 00000000
R138 (k110) = 00000000 R149 (k122) = 00000000 R154 (k124) = 00000000 R162 (k134) = 00000000
R139 (k111) = 00000000 R150 (k125) = 00000000 R155 (k126) = 00000000 R163 (k135) = 00000000
R140 (k112) = 00000000 R151 (k127) = 00000000 R156 (k128) = 00000000 R164 (k136) = 00000000
R141 (k113) = 00000000 R152 (k129) = 00000000 R157 (k130) = 00000000 R165 (k137) = 00000000
R142 (k114) = 00000000 R153 (k131) = 00000000 R158 (k132) = 00000000 R166 (k138) = 00000000
R143 (k115) = 00000000 R154 (k133) = 00000000 R159 (k134) = 00000000 R167 (k139) = 00000000
R144 (k116) = 00000000 R155 (k135) = 00000000 R160 (k136) = 00000000 R168 (k140) = 00000000
R145 (k117) = 00000000 R156 (k137) = 00000000 R161 (k138) = 00000000 R169 (k141) = 00000000
R146 (k118) = 00000000 R157 (k139) = 00000000 R162 (k130) = 0000000

How to use memory

1. **LOAD** from memory to register
 - `lw, lb, ld, ...` (`lw $t0, address`)
2. **COMPUTE** in registers
 - `add, ori, beq, jal,...` (`add $t2, $t0, $t1`)
3. **STORE** from register to memory
 - `sw, sb, sd,...` (`sw $t2, address`)

Addressing modes

Format	Address Computation
(register)	contents of register
imm	immediate
<u>imm (register)</u>	<u>immediate + contents of register</u>
symbol	address of symbol
symbol \pm imm	address of symbol + or - immediate
symbol \pm imm (register)	address of symbol + or - (immediate + contents of register)

Addressing modes

Loading from memory to \$t0: lw \$t0, **address?**

Imm+Register: (Only mode in bare machine)

la \$t1, **label** # load address of label to \$t1
lw \$t0, 2(\$t1) # address: address of label + 2

Immediate:

lw \$t0, 0x000AE430 # address: address 0x000AE430

Symbol:

lw \$t0, **label** # address: address of label

Register:

la \$t1, **label** # load address of label to \$t1
lw \$t0, \$t1 # address: address in \$t1

Symbol±Imm:

lw \$t0, **label+2** # address: address of label + 2

Symbol±Imm+Register:

lw \$t0, **label+2(\$t1)** # address: address of label + 2 + \$t1

```
.data
n: .word 0x2
m: .word 0x3
r: .space 4

.text
.globl main
main:
lw    $t0, n      # load n to $t0
lw    $t1, m      # load m to $t1
addu $t2, $t0, $t1 # $t2 ← ADD($t0, $t1)
sw    $t2, r      # store $t2 to r
```

```
.data
n: .word 0x2
m: .word 0x3
r: .space 4

.text
.globl main

main:
    la      $t5, n          # load address of n to $t5
    lw      $t0, 0($t5)      # load n to $t0

    la      $t5, m          # load address of m to $t5
    lw      $t1, 0($t5)      # load m to $t1

    addu   $t2, $t0, $t1    # $t2 ← ADD($t0, $t1)

    la      $t5, r          # load address of r to $t5
    sw      $t2, 0($t5)      # store $t2 to r
```

```
.data
n: .word 0x2, 0x3, 0x4

.text
.globl main

main:
    la      $t5, n          # load address of n to $t5
    lw      $t0, 0($t5)      # load n to $t0
    lw      $t1, 4($t5)      # load n+4 to $t1
    addu   $t2, $t0, $t1    # $t2 ← ADD($t0, $t1)
    sw      $t2, 8($t5)      # store $t2 to n+8
```

System calls

Service	System Call Code	Arguments	Result
print_int	1	\$a0 = integer	
print_float	2	\$f12 = float	
print_double	3	\$f12 = double	
print_string	4	\$a0 = string	
read_int	5		integer (in \$v0)
read_float	6		float (in \$f0)
read_double	7		double (in \$f0)
read_string	8	\$a0 = buffer, \$a1 = length	
sbrk	9	\$a0 = amount	
exit	10		address (in \$v0)

System calls – print_str

```
.data
str: .asciiz "Hello World"      # H,e,l,l,o, ,W,o,r,l,d,\0

.text
.globl main
main:
    li  $v0, 4                # code for print_str
    la  $a0, str               # argument
    syscall                   # executes print_str
```

System calls – read_int

```
.data
num: .space 4

.text
.globl main

main:
    li $v0, 5          # code for read_int
    syscall            # executes read_int
                      # return value is stored in $v0
    la $t0, num        # load address of num to $t0
    sw $v0, 0($t0)     # sw $v0, num
```

Branching

```
x ← read_int
```

```
y ← read_int
```

```
if x == y
```

```
    then print “Equal”
```

```
    else print “Not equal”
```

Branching

```
.text
.globl main
main:
    li $v0, 5
    syscall
    move $t0, $v0
    li $v0, 5
    syscall
    move $t1, $v0
    bne $t0, $t1, printNe
```

```
printEq:
    la $a0, strEq
    j print
```

```
printNe:
    la $a0, strNe
    j print
```

```
print:
    li $v0, 4
    syscall
```

```
.data
strEq: .asciiz "Equal"
strNe: .asciiz "Not equal"
```

Branching

```
.text
.globl main
main:
    li $v0, 5
    syscall
    move $t0, $v0

    li $v0, 5
    syscall
    move $t1, $v0

    seq $t2, $t0, $t1
    beq $t2, $0, printNe
```

```
printEq:
    la $a0, strEq
    j print

printNe:
    la $a0, strNe
    j print

print:
    li $v0, 4
    syscall

.data
strEq: .asciiiz "Equal"
strNe: .asciiiz "Not equal"
```

Looping

n ← read_int

counter ← 0
total ← 0

do
 counter ← counter + 1
 total ← total + counter
until counter == n

print total

Looping

```
.text
.globl main
main:
    li $v0, 5
    syscall

    move $t0, $v0

# $t0 is the original value

    li $t1, 0 # counter
    li $t2, 0 # sum
```

The diagram illustrates the control flow between the `main` function and a loop block. A curved arrow points from the `move $t0, $v0` instruction in `main` to the `loop:` label in the loop block. Another curved arrow points from the `j loop` instruction back to the `loop:` label, indicating the loop's body. The loop block contains the following assembly code:

```
loop:
    addi $t1, $t1, 1
    add $t2, $t2, $t1

    # counter = original value ?
    beq $t0, $t1, done
    j loop

done:
    li $v0, 1 # print_int
    move $a0, $t2
    syscall
```

Looping

```
.text
.globl main
main:
    li $v0, 5
    syscall

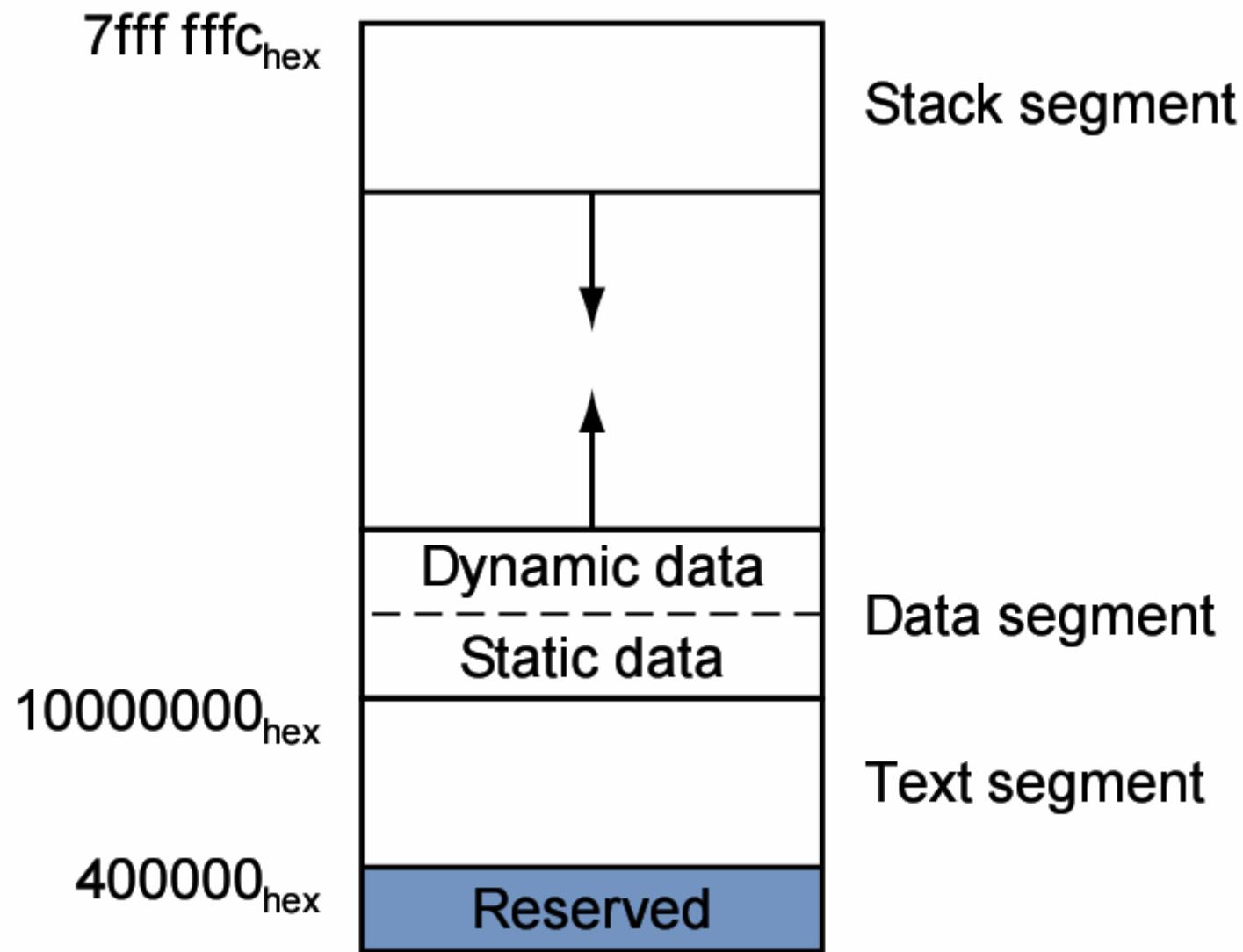
    move $t0, $v0
    # $t0 is the original value

    li $t1, 0 # counter
    li $t2, 0 # sum
```

```
loop:
    addi $t1, $t1, 1
    add $t2, $t2, $t1
    # counter = original value ?
    bne $t0, $t1, loop

done:
    li $v0, 1 # print_int
    move $a0, $t2
    syscall
```

Functions

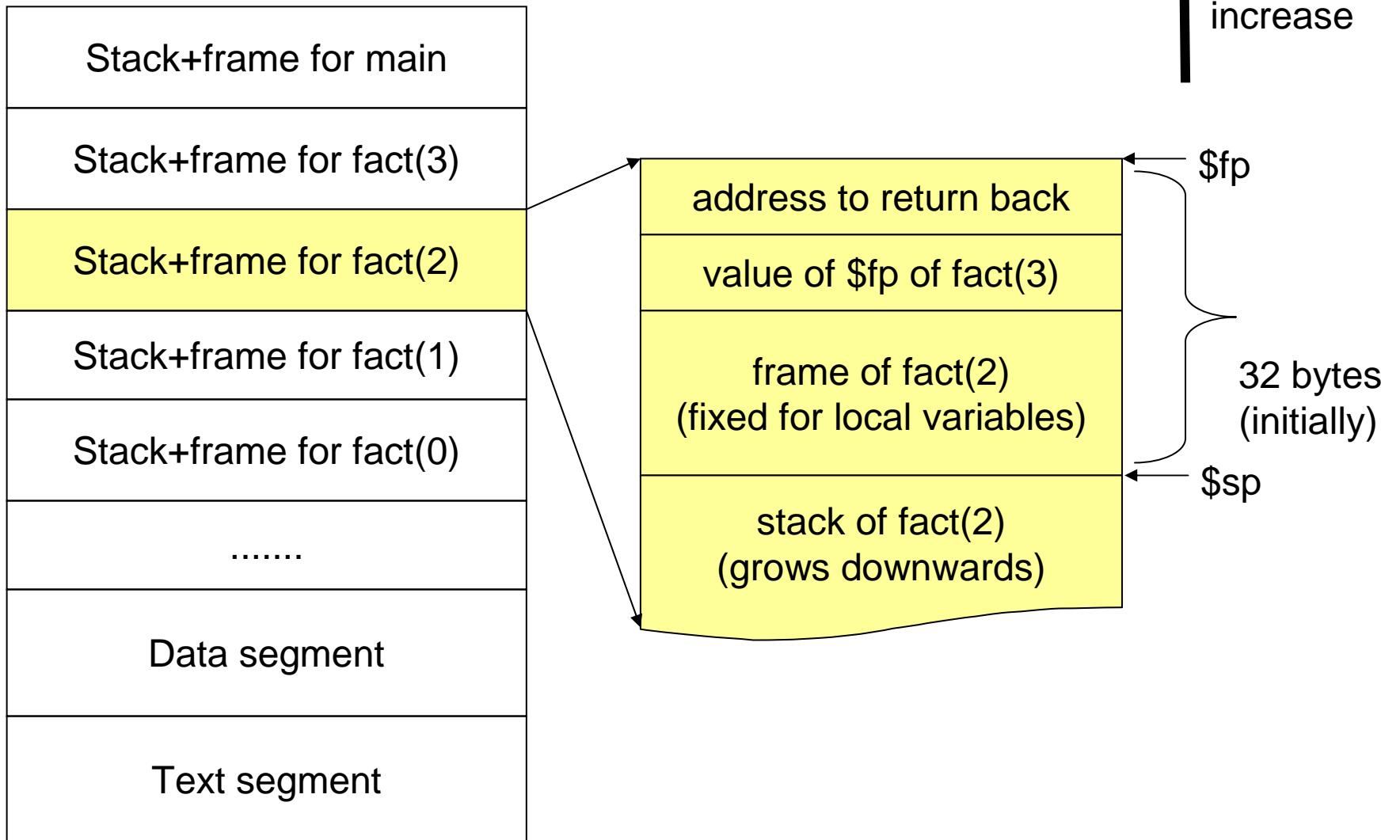


Factorial

```
main() {  
    x = fact(5);  
    ....  
    y = fact(6);  
}
```

```
fact(int n) {  
    if n == 0 or n == 1  
        then return 1;  
    else return n * fact(n-1);  
}
```

Factorial



Now it's your turn

- Write factorial program without functions
- Use branches and loops only