

RARS: RISC-V Assembler and Runtime Simulator

<https://github.com/TheThirdOne/rars>

Autore principale R. Giorgio di Unisi, modifiche D'Agostino/Delzanno

The RARS Project

RARS è un editor, assembler e simulatore dell'esecuzione di assembly e codice macchina per microprocessori RISC-V. RARS è un'applicazione Java application e quindi gira su qualsiasi OS. Richiede almeno Java 8 e si può scaricare dall'URL [home page](#). (le ultime release sono nel link a destra)

Scaricate ad esempio il file jar (*) rars1_5.jar sul vostro PC.

Aprite il file jar oppure eseguite il comando

```
java -jar rars1_5.jar
```

Su Linux potete anche creare lo script rars.sh:

```
#!/bin/bash java -jar rars1_5.jar
```

renderlo un file eseguibile con `chmod +x rars.sh`

e poi chiamare `./rars.sh`

RARS: Editor, registri e console

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

riscv1.asm

```
75 jalr x0,0(x1)
76
77 inizializza1:
78 li x5,0 # i=0
79 il_ciclo:
80 slli x6,x5,2 # i*4
81 add x7,x10,x6 # indirizzo di V[i]
82 li x8,1
83 sw x8,0(x7) # V
84 addi x5,x5,1
85 blt x5,x11,il_ciclo # salto a ciclo1 se i<dim
86 jalr x0,0(x1)
87
88 inizializza0:
89 mv x5,x10 # indirizzo di V[0]
90 slli x6,x11,2 # dim * 4
91 add x7,x10,x6 # indirizzo di V[dim]
92 i0_ciclo:
93 sw x0,0(x5) # memoria[p]=0
94 addi x5,x5,4 # p+8
95 bltu x5,x7,i0_ciclo # salta a ciclo 2 se p<&V[dim]
96 jalr x0,0(x1)
97
98
99
```

Line: 99 Column: 2 ☒ Show Line Numbers

Messages Run I/O

Assemble: assembling /private/var/folders/2m/8c0190kj4n93sz7fj15kn4hm0000gn/T/hesperdata_giorgiodelzanno/riscv1.asm

Clear

Assemble: operation completed successfully.

Name	Number	Value
zero	0	0
ra	1	0
sp	2	2147479548
gp	3	268468224
tp	4	0
t0	5	0
t1	6	0
t2	7	0
s0	8	0
s1	9	0
a0	10	0
a1	11	0
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	0
s2	18	0
s3	19	0
s4	20	0
s5	21	0
s6	22	0
s7	23	0
s8	24	0
s9	25	0
s10	26	0
s11	27	0
t3	28	0
t4	29	0
t5	30	0
t6	31	0
pc		4194304

Assemblatore e memoria

/private/var/folders/2m/8c0190kj4n93sz7fj15kn4hm0000gn/T/hspferdata_giorgiodelzanno/riscv1.asm - RARS 1.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x0fc10517	auipc x10,64528	41: la x10,V
<input type="checkbox"/>	0x00400004	0x00050513	addi x10,x10,0	
<input type="checkbox"/>	0x00400008	0x00800593	addi x11,x0,8	42: li x11,8
<input type="checkbox"/>	0x0040000c	0x050000ef	jal x1,40	43: jal x1,stampav
<input type="checkbox"/>	0x00400010	0x0fc10517	auipc x10,64528	45: la x10,V
<input type="checkbox"/>	0x00400014	0xff050513	addi x10,x10,-16	
<input type="checkbox"/>	0x00400018	0x00800593	addi x11,x0,8	46: li x11,8
<input type="checkbox"/>	0x0040001c	0x0c8000ef	jal x1,100	47: jal x1,inizializza0
<input type="checkbox"/>	0x00400020	0x0fc10517	auipc x10,64528	49: la x10,V

Labels

Label	Address
(global)	
main	0x00400000
riscv1.asm	
stampav	0x00400005c
sV_ciclo	0x004000080
inizializza1	0x0040000c4
i1_ciclo	0x0040000c8
inizializza0	0x0040000e4

☒ Data ☒ Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	2	3	4	5	2	3	2	10
0x10010020	8	7	2	3	4	5	2	3
0x10010040	655392	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0

☒ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

0x10010000 (.data)

Messages Run I/O

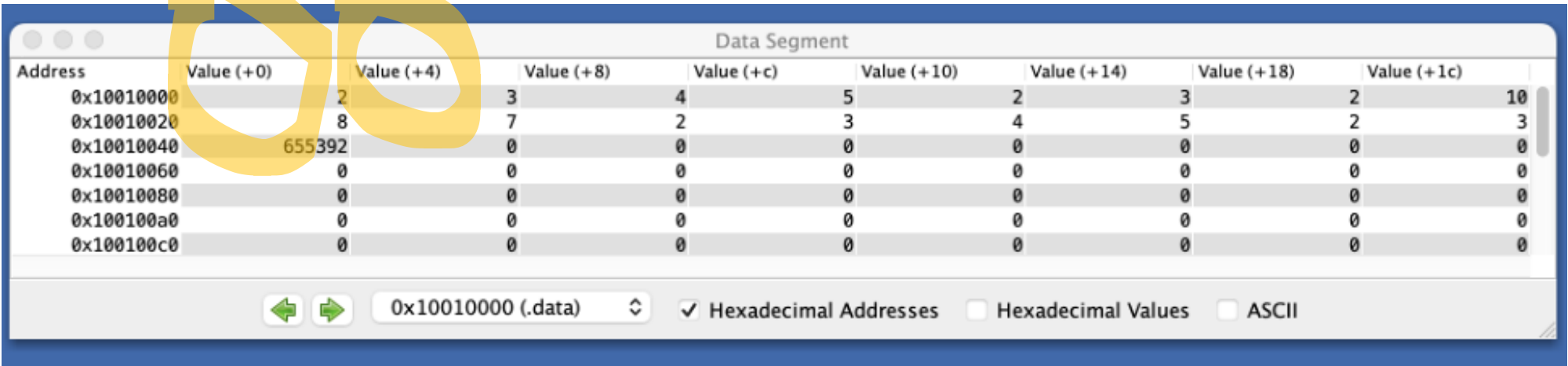
Assemble: assembling /private/var/folders/2m/8c0190kj4n93sz7fj15kn4hm0000gn/T/hspferdata_giorgiodelzanno/riscv1.asm

Assemble: operation completed successfully.

Registers

Name	Number	Value
zero	0	0
ra	1	0
sp	2	2147479548
gp	3	268468224
tp	4	0
t0	5	0
t1	6	0
t2	7	0
s0	8	0
s1	9	0
a0	10	0
a1	11	0
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	0
s2	18	0
s3	19	0
s4	20	0
s5	21	0
s6	22	0
s7	23	0
s8	24	0
s9	25	0
s10	26	0
s11	27	0
t3	28	0
t4	29	0
t5	30	0
t6	31	0
pc		4194304

RARS: memoria con allineamento alla parola (4 byte)



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	2	3	4	5	2	3	2	10
0x10010020	8	7	2	3	4	5	2	3
0x10010040	655392	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0

Navigation: ← → 0x10010000 (.data) ⚙

Options: ☒ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

INDIRIZZI CONSECUTIVI DI PAROLE
(ALLINEAMENTO ALLA PAROLA)

RARS: Esecuzione e I/O

/private/var/folders/2m/8c0190kj4n93sz7fj15kn4hm0000gn/T/hsperrdata_giorgiodelzanno/riscv1.asm - RARS 1.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

riscv1.asm

```
75 jalr x0,0(x1)
76
77 inizializza1:
78 li x5,0 # i=0
79 i1_ciclo:
80 slli x6,x5,2 # i*4
81 add x7,x10,x6 # indirizzo di V[i]
82 li x8,1
83 sw x8,0(x7) # V
84 addi x5,x5,1
85 blt x5,x11,i1_ciclo # salto a ciclo1 se i<dim
86 jalr x0,0(x1)
87
88 inizializza0:
89 mv x5,x10 # indirizzo di V[0]
90 slli x6,x11,2 # dim * 4
91 add x7,x10,x6 # indirizzo di V[dim]
92 i0_ciclo:
93 sw x0,0(x5) # memoria[p]=0
94 addi x5,x5,4 # p+8
95 bltu x5,x7,i0_ciclo # salta a ciclo 2 se p<&V[dim]
96 jalr x0,0(x1)
97
98
99
```

Line: 99 Column: 2 ☒ Show Line Numbers

Messages Run I/O

Clear

```
8
1 1 1 1 1 1 1 1
-- program is finished running (0) --
```

Registers

Name	Number	Value
zero	0	0
ra	1	0
sp	2	2147479548
gp	3	268468224
tp	4	0
t0	5	0
t1	6	0
t2	7	0
s0	8	0
s1	9	0
a0	10	0
a1	11	0
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	0
s2	18	0
s3	19	0
s4	20	0
s5	21	0
s6	22	0
s7	23	0
s8	24	0
s9	25	0
s10	26	0
s11	27	0
t3	28	0
t4	29	0
t5	30	0
t6	31	0
pc		4194304

File Edit Run Settings Tools Help

Run speed at max (no interaction)

✓ Show Labels Window (symbol table)
Program arguments provided to program
Popup dialog for input syscalls (5,6,7,8,12)
✓ Addresses displayed in hexadecimal
Values displayed in hexadecimal

Assemble file upon opening
Assemble all files in directory
Assemble all files currently open
Assembler warnings are considered errors
✓ Initialize Program Counter to global 'main' if defined

✓ Permit extended (pseudo) instructions and formats
Self-modifying code
✓ 64 bit

Editor...
Highlighting...
Exception Handler...
Memory Configuration...

Bkpt Address

0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000
0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000
0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000
0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000
0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000
0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000
0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000
0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000
0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000
0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000	0x0040000

Labels

Label	Address
(global)	
main	0x00400000
riscv1.asm	
stampaV	0x0040005c
sV_ciclo	0x00400080
inizializza1	0x004000c4
i1_ciclo	0x004000c8
inizializza0	0x004000e4

✓ Data ✓ Text

Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)

0x10010000	2	3	4	5	2	3	2	10
0x10010020	8	7	2	3	4	5	2	3
0x10010040	655392	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0

RV64

STEP BY STEP

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Assemble F3
Go F5
Step F7
Backstep F8
Pause F9
Stop F11
Reset F12

Clear all breakpoints %K
Toggle all breakpoints %T

0x00400028 0x00800593 addi x11,x0,8
0x0040002c 0x030000ef jal x1,24
0x00400030 0x0fc10517 auipc x10,64528
0x00400034 0xfdf05013 addi x10,x10,-48
0x00400038 0x00800593 addi x11,x0,8
0x0040003c 0x088000ef jal x1,68

Text Segment

Source	
47: jal x1,inizializza0	
49: la x10,V	
50: li x11,8	
51: jal x1,stampaV	
53: la x10,V	
54: li x11,8	
55: jal x1,inizializza1	

Labels

Label	Address
(global)	
main	0x00400000
riscv1.asm	
stampaV	0x0040005c
sV_ciclo	0x00400080
inizializza1	0x004000c4
i1_ciclo	0x004000c8
inizializza0	0x004000e4

✓ Data ✓ Text

Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)

0x10010000	2	3	4	5	2	3	2	10
0x10010020	8	7	2	3	4	5	2	3
0x10010040	655392	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0

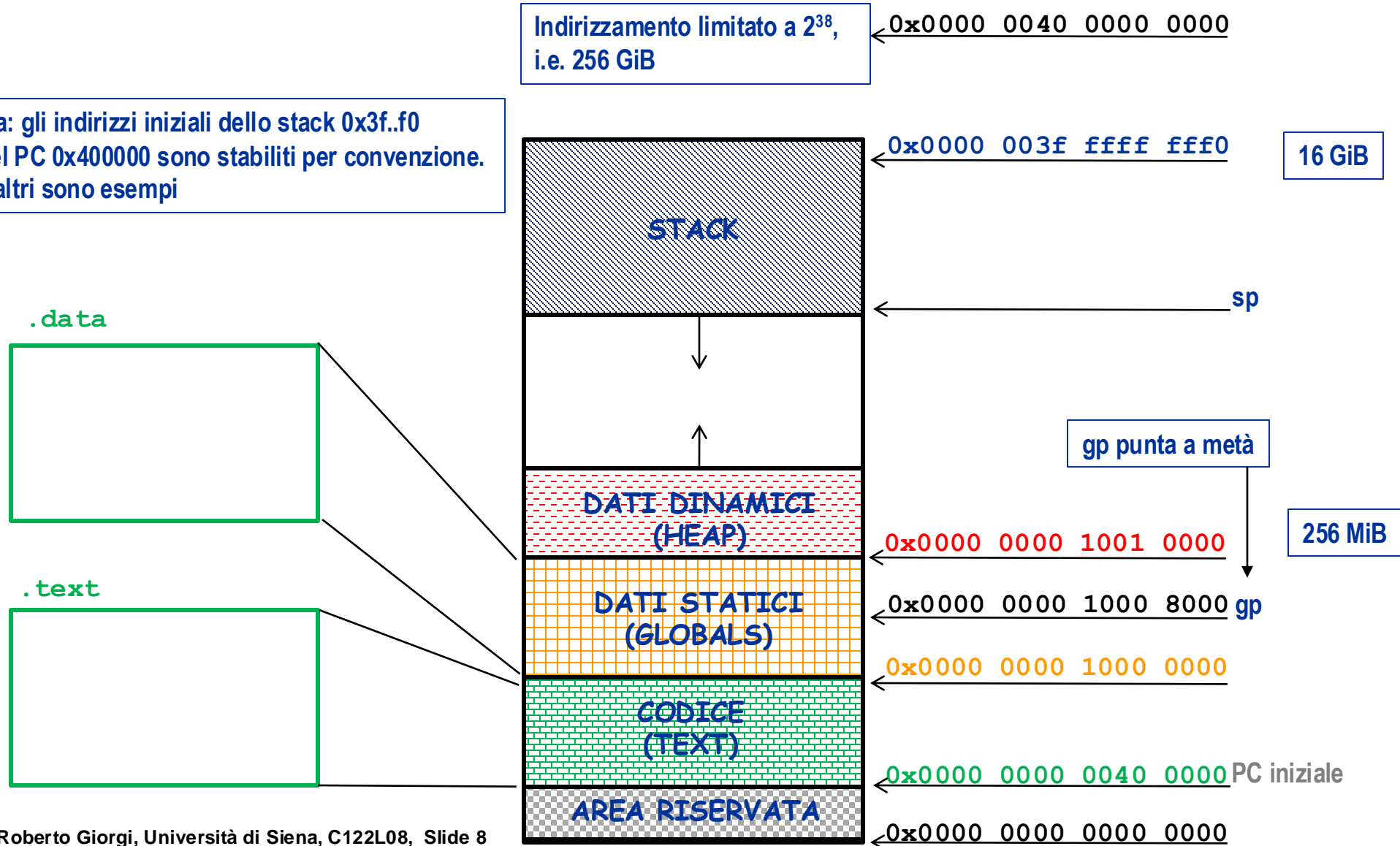
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

NEXT BREAKPOINT

Mappa di memoria (Memory Layout)

- Dove si trova lo stack? il programma ? i dati globali ?

Nota: gli indirizzi iniziali dello stack 0x3f..f0 e del PC 0x400000 sono stabiliti per convenzione. Gli altri sono esempi



Principali direttive per l'assemblatore

- .data [<addr>]** marca l'inizio di una zona dati;
se <addr> viene specificato, i dati sono memorizzati
a partire da tale indirizzo <addr>
- .text [<addr>]** marca l'inizio del codice assembly;
se <addr> viene specificato, i dati sono memorizzati
a partire da tale indirizzo <addr>
- .globl <symb>** dichiara un simbolo <symb> come visibile dall'esterno
(gli altri simboli sono locali per default) in modo che
possa essere usato da altri file
- .extern <symb> [<size>]** dichiara che il dato memorizzato all'indirizzo
<symb> è un simbolo globale ed occupa <size> byte.
Questo consente all'assemblatore di:
 - 1) memorizzare il dato in una porzione del segmento
dati (quello indirizzato tramite il registro gp)
 - 2) dichiarare che i riferimenti al simbolo <symb>
riguardano un oggetto esterno ad un modulo
(torneremo su questo fra poco)
- .asciz "str"** mette in memoria la stringa "str", seguita da un byte '0'
- .ascii "str"** mette in memoria la stringa "str", SENZA lo '0' finale
- .string "str"** mette in memoria la stringa "str", simile a asciz

Ulteriori direttive assemblatore

.space <n>

Mette in memoria n spazi (ovvero riserva n byte di memoria)

.byte <b1>, ..., <bn>

Mette in memoria gli n byte che sono specificati da <b1>, ... , <bn>

.word <w1>, ..., <wn>

Mette in memoria le n word a 32-bit che sono specificate da <w1>, ... , <wn>

.float <f1>, ..., <fn>

Mette in memoria gli n numeri floating point a singola precisione (32 bit)
(in locazioni di memoria contigue)

.double <d1>, ..., <dn>

Mette in memoria gli n numeri floating point a doppia precisione (64 bit)
(in locazioni di memoria contigue)

.half <h1>, ..., <hn>

Mette in memoria le n quantità a 16 bit (in locazioni di memoria contigue)

.align <n>

Allinea il dato successivo a un indirizzo multiplo di 2^n . Es. `.align 2` allinea il valore successivo "alla word". `.align 0` disattiva l'allineamento generato dalle direttive `.half`, `.word`, `.float`, e `.double` fino alla successiva direttiva `.data` o `.kdata`

Servizi di sistema (system call) – istruzione 'ecall'

- 'ecall' serve per chiedere un servizio al sistema operativo (operazione nota come 'system call' attraverso una 'trap')
 - Il numero del servizio è indicato in **a7***
 - 1: stampa un intero a video
a0 in ingresso, contiene l'intero in binario da stampare (*)
 - 4: stampa un messaggio a video
a0 in ingresso, indirizzo della stringa da stampare (*)
 - 5: leggi un intero dalla tastiera
a0 in uscita, conterrà l'intero letto in binario (*)
 - 10, 93: termina il programma
il sistema operativo riprende il controllo del calcolatore
nella versione con codice 93 si può passare codice (es 0, -1...) nel parametro **a0**

Nota: si può assumere che la ecall si comporti come una chiamata a funzione speciale in cui non deve essere alterato alcun registro (salvo quelli utilizzati in ingresso o uscita naturalmente).

(*) Con riferimento al simulatore «RARS»; altri simulatori usano rispettivamente a0 e a1 anziché a7 e a0

Altre direttive utili

`.eqv DEF value`

definisce DEF come value

`.macro nome (%arg1, ..., %argn)`



`.end macro`

definisce macro con argomenti
(%argi anche nel corpo)

Esempi di macro ed ecall

```
.macro print_str (%X)
    li a7, 4
    la a0, %X
    ecall
.end_macro
```

Carico codice sys_call

```
.macro print_int (%X)
    li a7, 1
    mv a0, %X
    ecall
.end_macro
```

Carico l'argomento di tipo
indirizzo in a0 (la = load address)

```
.macro scan_int
    li a7, 5
    ecall
.end_macro
```

In questo caso il parametro sarà
l'etichetta della stringa allocata
nel data segment (indirizzo
iniziale della sequenza di byte
con i diversi caratteri)

```
.macro exit (%X)
    li a7, 93
    li a0, %X
    ecall
.end_macro
```

Esempio Hello world senza macro

```
.globl main

.data
msg: .string "\nHello word"

.text
main:
# stampa la stringa memorizzata
# a partire da indirizzo "msg"

li a7,4
la a0,msg
ecall

#esci con codice 0
li a7,10
ecall
```

Esempio Hello world con macro

```
.globl main
.data
msg: .string "\nHello word"

.macro print_str (%x)
    li a7, 4
    la a0, %x
    ecall
.end_macro

.macro print_int (%x)
    li a7, 1
    mv a0, %x
    ecall
.end_macro

.macro scan_int
    li a7, 5
    ecall
.end_macro

.macro exit (%x)
    li a7, 93
    li a0, %x
    ecall
.end_macro

.text
main:
print_str(msg)
exit(0)
```

Esercizio - somma_vettore_nofor

.globl main

.eqv DIM 10

.data

V: .word 1,2,3,4,5,6,7,8,9,10

b: .string "The sum is "

.macro ...

//assumiamo la definizione delle macro che ci servono

.text

main:

//Calcolate la somma degli elementi del vettore in S0.

//Poi provate a ridefinirlo come vettore di char e modificate il programma

Deve stampare: The sum is 55

print_int(s0)

exit(0)