INTRODUCCIÓ A L'ENTORN DE LABORATORI D'ESC1

Treball en assemblador de SISA-I i SISA-F

Els passos habituals per fer un programa en assemblador són els següents: primer es crea el programa escrivint-lo en aquest llenguatge assemblador per mitjà d'un editor de programes. El resultat és un fitxer font en un llenguatge que pot entendre l'usuari, però no la màquina. Per traduir-lo a llenguatge màquina es fa servir un programa traductor, també anomenat assemblador. Aquest genera un fitxer objecte amb la traducció del vostre programa. Molts programes estan formats per múltiples fitxers font que es poden traduir per separat en múltiples fitxers objecte. Per tal de crear el fitxer executable amb el programa final cal encara un darrer pas. S'anomena enllaçat o muntatge (*link*) i es realitza amb un programa enllaçador o muntador (*linker*).

Durant el procés de creació d'un programa se solen produir errors. Hi ha dos tipus d'errors: els sintàctics o detectables en temps de traducció, i els semàntics o detectables en temps d'execució. Els errors sintàctics són, per exemple, escriure malament una instrucció o fer una operació entre dos tipus de dades incompatibles. Aquests errors són detectats pel traductor i els hem de solucionar abans de poder generar un executable. Un cop tenim un programa sintàcticament correcte el podem executar, però això no implica que el programa sigui correcte. Totes les instruccions poden ser correctes, però ens podem haver oblidat de posar la condició de final d'un bucle (i que aquest no acabi mai) o que, senzillament, el programa no faci el que nosaltres volem. Aquests errors només es poden detectar en temps d'execució, i per tal d'eliminar-los fem servir un depurador de programes (*debugger*). Els depuradors ens permeten executar el programa instrucció a instrucció i veure tots els valors que es van calculant, de manera que podem trobar els errors.

L'entorn de desenvolupament dels nostres programes serà sota **Linux**. Farem servir un editor qualsevol de Linux (vi, gedit, xedit, kwrite, etc.). L'assemblador s'anomena **sisa-as**, el muntador s'anomena **sisa-ld**, i el programa simulador que permet l'execució i/o depuració dels executables es diu **sisa-dbg**.

Treball previ: Instal·lació / configuració de software

Les pràctiques es realitzaran usant un ordinador amb Linux. Es poden fer als ordinadors de l'EPSEVG, iniciant-los carregant una imatge Linux, o bé usant un equip propi en el que hi tingueu instal·lat un sistema Linux (natiu o màquina virtual).

- **Configuració del software a les aules de laboratori de l'EPSEVG**

En cas que useu els ordinadors de l'EPSEVG, en aquesta primera sessió de pràctiques configurareu el software necessari per a realitzar les pràctiques:

1. Entra al Linux en els ordinadors de l'aula de pràctiques.
2. Executa la comanda següent:
    - `export PATH=/opt/ec1/bin:$PATH`
    - O bé, si vols evitar haver d'executar aquesta comanda cada vegada, edita el fitxer `.bashrc` del teu «home» i afegeix-la al final de tot.
3. Ara ja pots usar els programes que necessites per fer les pràctiques, que es troben en el directori `/opt/ec1/bin`. Prova el correcte funcionament de la configuració executant la comanda `sisa-dbg` des de qualsevol directori d'usuari.

- **Instal·lació i configuració del software al teu ordinador personal**

Alternativament, si disposes d'un equip propi amb sistema Linux i vols instal·lar el software de l'assignatura, hauràs de seguir els passos següents:

1. Entra en Linux en el teu ordinador.
2. Descarrega't des d'Atenea els fitxers:
    - `simulador_sisa.tar.gz`
    - `install_sisa.sh`
3. Deixa els dos fitxers al directori de treball que vulguis (és on s'instal·larà el simulador) i executa l'script `install_sisa.sh` amb la comanda:
    - `source install_sisa.sh`
4. L'execució de l'script descomprimeix el simulador al directori actual i modifica les variables d'entorn `PATH` i `LD_LIBRARY_PATH`. Per comprovar que el simulador s'ha instal·lat correctament, executa la comanda `sisa-dbg`

És possible que el teu ordinador sigui d'arquitectura de 64 bits. Si és el cas, hauràs d'instal·lar també unes llibreries per compatibilitzar-lo amb aquest software de 32 bits:

1. *sudo dpkg --add-architecture i386*
2. *sudo apt-get update*
3. *sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386*

A continuació, prova el correcte funcionament de la instal·lació executant la comanda `sisa-dbg` i comprova que s'executa el simulador correctament.

PRÀCTICA 0

1. Edició dels fitxers

L'edició de fitxers la farem amb un editor qualsevol que ens permeti generar text pla, ja sigui en mode text (vi, joe) o gràfic (gedit, kwrite, xedit, etc). Exemples:

```
$ gedit s0.s &
$ vi s0.s
```

Si activem un editor gràfic com en el primer cas, el signe **&** al final de la línia de comandes evitarà que l'editor bloquegi la finestra de comandes (el mateix efecte s'obtindria activant l'editor des d'un menú o bé clicant la seva icona). En el segon cas, l'editor **vi** ocuparà la finestra fins que en sortim, però podem obrir altres finestres de comandes simultàniament.

Exemple de programa en assemblador sisa-i

```
.include "macros.s"

N = 10

.data

RES:  .word  0
V:    .word  -1, -2, -3, -4, -5, -6, -7, -8, -9, -10

.text

main:
      MOVI        R0, 0           ; Posem R0 a 0
      MOVI        R1, 0           ; Posem R1 a 0
      MOVI        R2, N*2         ; Límit del comptatge
      MOVI        R3, 0           ; Suma parcial
      $MOVEI      R4, V           ; Adreça inicial de V
bucle:
      CMPLT       R5, R1, R2
      BZ          R5, fibucle     ; Després de 10 voltes acaba
      ADD         R6, R4, R1
      LD          R5, 0(R6)       ; Llegim un element
      ADD         R3, R3, R5      ; Acumulem
      ADDI        R1, R1, 2       ; Cada element ocupa 2 bytes!
      BZ          R0, bucle
fibucle:
      $MOVEI      R4, RES         ; Adreça de RES
      ST          0(R4), R3
      HALT
```

Exercici 1 (per lliurar)

1) Què fa aquest programa?

2) Quin és el valor final de RES?

3) A quina posició de memòria s'escriu al final del programa?

2. Assemblatge, muntatge i depuració

La generació de codi màquina a partir del codi assemblador la farem amb el programa assemblador executant la comanda següent:

```
sisa-as --gstabs+ -o s0.o s0.s
```

Aquesta comanda assemblarà el fitxer de codi font `s0.s`, i generarà un fitxer de codi objecte, anomenat `s0.o` En aquesta instrucció, l'opció `--gstabs+` serveix per incloure informació de depuració, l'opció `-o s0.o` serveix per indicar quin nom ha de tenir el fitxer objecte resultant, i l'argument `s0.s` és el nom del nostre fitxer que volem assemblar. A continuació, cal enllaçar o muntar aquest fitxer objecte per crear el fitxer executable:

```
sisa-ld -o s0 s0.o
```

L'opció `-o s0` indica quin nom ha de tenir el fitxer executable resultant. Cal tenir en compte que el procés d'assemblatge pot generar errors. Sempre que ens trobem amb un error, el que haurem de fer és fixar-nos en el comentari que l'acompanya i anotar el número de línia on s'ha detectat. Llavors cal intentar corregir-lo editant novament el fitxer i tornant a assemblar el programa. Això és un procés iteratiu fins que no s'obtinguin errors.

Per exemple, editeu el fitxer `s0.s`, i canvieu la instrucció `MOVI R2,N*2` per `MOVI R8,N*2`. A continuació torneu a assemblar el programa i observeu el missatge d'error. Ara editeu-lo de nou, corregiu l'error, i torneu-lo a assemblar i muntar. El fitxer `s0` conté la traducció a llenguatge màquina del codi que s'ha mostrat anteriorment. Aquest fitxer ja es podrà carregar directament al depurador.

El depurador (*debugger*) s'anomena **sisa-dbg**. És convenient maximitzar la finestra de terminal abans d'invocar-lo, per tenir més informació a la vista. El podeu invocar sense arguments o posant-li com a argument el fitxer que voleu executar:

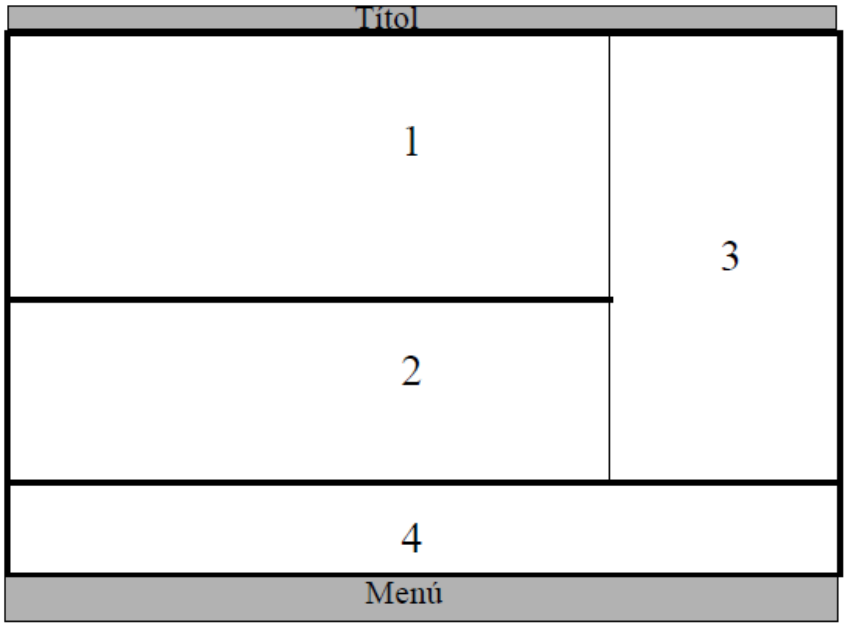
```
sisa-dbg s0
```

A continuació s'explicaran breument les comandes més usuals del depurador. Totes les comandes del depurador s'han d'introduir amb el teclat (no amb el ratolí!). La majoria estan disponibles en dos menús: un de general que s'activa/desactiva amb la tecla **F12**, i un altre de particular per a cada panell que s'activa/desactiva amb la tecla **F11** (o bé amb la tecla **?** en alguns terminals, o també la tecla **'**). En cada menú, que pot oferir diversos nivells de submenús, se seleccionen les opcions amb la tecla **INTRO**. Es pot abandonar sense seleccionar res prement la mateixa tecla **F11** (o bé ?, o bé ')/ **F12** amb què s'ha entrat. No obstant, les comandes més usades del depurador estan disponibles també amb una sola pulsació (curtcircuit), ja sigui amb les tecles **F1**-**F10**, o bé amb altres tecles que anirem coneixent.

Per exemple, prement la tecla **F1** apareixerà una pantalla **d'ajuda** informant-nos de totes les comandes disponibles, amb una descripció breu. Per sortir del depurador premeu **F12->File->Exit,** o bé simplement **Ctrl-X**.

3. Panells i vista del depurador

L'aparença que obtindrem inicialment per pantalla es mostra a la següent figura, i està formada per 4 *panells* o finestres.



Cada panell mostra una *vista*, o tipus d'informació. En general, per tal d'operar canvis en algun dels panells, cal enfocar-lo primer (seleccionar-lo) prement la tecla **TAB** repetidament fins que el seu títol queda destacat en vídeo invers.

Exercici 2 (per lliurar)

Proveu, per exemple, de prémer **TAB** un cop i seleccionar el panel 2 (Vista de Dades). A continuació moveu els cursors amunt i avall per examinar qualsevol part de la memòria. Seleccioneu els altres panells, i contesteu la pregunta següent:

- Quin tipus d'informació creieu que mostra cada un dels 4 panells?

4. Vista del codi desassemblat i vista del codi font

La Vista del codi font es troba inicialment al panell 3, i mostra el contingut del fitxer font a partir del qual hem generat el programa executable, amb els comentaris i les macros.

La vista del codi desassemblat es troba inicialment al panell 1, i mostra una porció de la memòria desassemblada (en binari), una instrucció per línia. Cada línia està composta de tres columnes. A la primera columna apareixen símbols del programa: seccions com `<.text>` o `<.data>`, etiquetes com `<bucle>`, `<fibucle>`, constants com `<N>` (que mostra erròniament com si fos una etiqueta), i variables com `<V>` (la variable `<RES>` no apareix perquè coincideix amb la mateixa adreça que la secció `<.data>`, i l'etiqueta `<main>` tampoc perquè coincideix amb `<.text>`). A la segona columna apareixen les adreces de memòria en hexadecimal. A la tercera columna hi ha el contingut de memòria en hexadecimal (el codi de cada instrucció), i a continuació la instrucció en

assemblador. Per a algunes instruccions pot aparèixer al final, en forma de comentari, el valor en hexadecimal d'un dels operands (etiquetes, immediats, etc).

Exercici 3

Enfoqueu el panell 1. Recorreu la memòria polsant les tecles de cursor amunt i avall. Veureu que en aquelles regions de memòria que continguin dades en lloc de codi, el que es mostra no tindrà gaire sentit.

La memòria és molt gran, i si heu perdut el punt on estàveu, podeu retornar-hi amb les tecles **A** o **G**. La tecla **A** (anar a posició "Actual") situa el cursor a l'adreça indicada pel registre **PC**. La tecla G ("Go") situa el cursor sobre l'adreça que li indiquem. Per exemple, polseu G, i quan us demani l'adreça escriviu: `bucle` o bé `main`.

5. Vista de dades de memòria

Es troba per defecte en el panell 2, mostra el contingut d'una porció de la memòria en un format compacte: en cada línia de pantalla es mostra primer una adreça, i a continuació les dades emmagatzemades en aquesta i en les següents adreces de memòria, fins a omplir la línia de pantalla. Per tant, l'adreça mostrada en primer lloc correspon tan sols a la primera de les dades de la línia. El nombre de dades mostrades per línia depèn del format de visualització elegit. Al final de cada línia apareix per segon cop el contingut de les mateixes posicions, però en format ASCII (tan sols els valors que corresponen a caràcters tipogràfics, els altres apareixen en blanc).

Aquesta vista es pot mostrar en els següents formats:

- ASCII. Mostra byte per byte, sense espais entremig, en format de caràcters.
- Hexadecimal, en mida byte: els valors de cada byte en hexadecimal
- Hexadecimal, en mida word: una sola dada per a cada parell de posicions de memòria.

Recordeu que els word s'emmagatzemen en format *little endian*, i per tant els seus dígits apareixen en ordre invers a com es veuen en el format byte (però el contingut de la memòria no canvia, és sols la manera com ens ho presenta el depurador!).

- Hexadecimal, en mida long: una sola dada en hexadecimal per a cada quatre posicions de memòria, segons el conveni *little endian*.
- Decimal entera amb signe, ja sigui en mida byte o word. Per veure enters.
- Decimal entera sense signe, ja sigui en mida byte o word. Per veure naturals.
- Decimal fraccionària, ja sigui en notació exponencial o de punt fix. Per veure reals codificats en coma flotant (word).

Exercici 4

Enfoqueu la vista de dades (**TAB**) i premeu repetidament la tecla **F** per canviar el format de visualització per un dels 9 formats possibles. Proveu també amb el menú **F11**(o bé **?** o bé **'**)**->Format->Dec->SignedWord**. Ara podreu observar a l'adreça 0x0022

la variable suma (val 0) i, a continuació, a l'adreça 0x0024 i següents, els elements enters del vector **V** (valors -1, -2, ...).

6. Vista de registres

Es troba per defecte en el panell 4. Mostra el contingut dels registres R0 ... R7, de S0 ... S7 i del PC, tots en hexadecimal, així com dels registres F0 ... F7, en decimal i coma fixa.

7. Vista de guaites i altres vistes

La configuració inicial mostra la vista de codi desassemblat ("Disasm") al panell 1, la vista de dades ("Data") al panell 2, la vista de codi font ("Source") al panell 3, i la vista de registres ("Registers") al panell 4. Sols hi ha 4 panells, però en cada un podem posar-hi fins a 7 vistes diferents, a triar entre les ja explicades o bé una de les següents: Traça ("Trace") i Guaites ("Watches"). Per canviar la vista associada a un panell, premeu la següent opció de menú: **F12->Pannels->***Núm del panell***->***Nom de la vista*.

- ***Traça***: mostra una llista ordenada de les instruccions executades.
- ***Guaites***: mostra el valor actualitzat de les variables que vulguem monitoritzar. Inicialment no conté cap variable. També podem indicar-ne el format.
- També podem tancar el panell ("***Close***"), i tenir més espai per als altres panells.

Exercici 5

A continuació mostrarem la vista de guaites al panell 2 i hi afegirem dues guaites per monitoritzar les variables **RES** i **V**. Seguim els següents passos:

- Associem la vista de Guaites al panell 2: **F12->Pannels->2->Watch**
- Enfoquem el panell 2 amb **TAB**
- **F11(o bé ? o bé ')->Add->Format->Dec->SignedWord->1->RES**
- **F11(o bé ? o bé ')->Add->Format->Dec->SignedWord->10->V**

Ara podrem moure el cursor amunt i avall i també a dreta i esquerra, per veure els elements de V si no hi caben al panell.

Al igual que en la vista de dades, les guaites actualitzen el seu valor a mesura que executem el programa. Per a cada guaita es mostren 3 informacions:

- A la primera columna, l'expressió introduïda per nosaltres (per exemple, **RES**). L'expressió pot contenir números, constants simbòliques i operadors en el format usual del llenguatge C. També pot contenir símbols del programa com ara etiquetes i variables, i fins i tot noms de registres.
- A la segona columna, el valor numèric equivalent de l'expressió, entre parèntesis i en hexadecimal. Quan el depurador avalua l'expressió, substitueix les etiquetes i variables per les seves adreces equivalents. Els registres se substitueixen pel seu contingut actual.

- A continuació, el valor emmagatzemat a memòria a l'adreça calculada en la segona columna i a les adreces següents, tants elements com li hàgim indicat, en el format indicat.

Quan afegim una nova guaita, un cop introduït el format i el nombre d'elements, ens demana una expressió que determini l'adreça inicial a mostrar. Ha de ser una expressió matemàtica que pot incloure operadors (+, -, *, /, parèntesis), números, o bé símbols del programa com etiquetes o constants. Però NO ADMET indexats (p.ex. V[3]). Així, si volem que ens mostri V[3], hem de posar: V+2*3.

Exercici 6 (per lliurar)

Quina expressió hauríeu d'introduir per mostrar únicament l'element V[N-1]? Proveu-ho afegint una guaita i comprovant que val -10.

8. Execució de programes

Aquesta és la funcionalitat més important del depurador. Hi ha dues maneres principals d'executar un programa: execució pas a pas, o bé, execució contínua amb possibles punts d'aturada.

a) **Execució pas a pas**: El **PC** sempre conté l'adreça de la següent instrucció a executar, la qual està marcada amb un petit signe ***** a la dreta de la seva adreça (punt d'execució). Prement la tecla **F5** s'executa la instrucció apuntada pel **PC** i s'actualitza automàticament el valor dels registres i dades de memòria en tots els panells. Si s'ha fet un accés a la memòria, la posició accedida queda ressaltada en vídeo invers per facilitar el control visual, i si s'ha modificat un registre, aquest també queda ressaltat. Aquest és el mètode més útil per a programes petits com els que nosaltres escriurem.

Exercici 7

Executeu pas a pas tot el programa prement **F5**, i observeu a la vista de dades com es ressalten els elements de **V** i la variable **RES** a mesura que són llegits o escrits. A la vista de registres podeu observar com es van modificant.

b) **Execució contínua**: Prement la tecla **F4**, el programa s'executarà fins al final (o fins al primer punt d'aturada que trobi, segons veureu al següent apartat). És el mètode més útil per veure ràpidament els resultats d'un programa llarg, suposant que no té errors.

Exercici 8

Premeu **F9** per recarregar el programa i executeu-lo fins al final amb **F4**. Comproveu el valor final de la variable **RES** (a l'adreça 0x0022 de la vista de dades, o bé en una guaita), aquest ha de ser -55.

c) **Execució contínua amb punts d'aturada**: Sovint, els programes tenen errors i cal executar-los pas a pas per descobrir l'error. Però si són molt llargs, resulta pesat i és molt més pràctic executar-los de forma contínua, però aturant-nos en algun punt clau. Aquest punt rep el nom de punt d'aturada, i és una marca situada en una instrucció perquè el programa s'aturi abans d'executar-la. Per

afegir un punt d'aturada sols cal enfocar la vista de codi desassemblat, situar el cursor sobre la instrucció en qüestió, i polsar la tecla **B** ("breakpoint") o bé **INTRO**. Apareixerà un signe **+** a l'esquerra de la instrucció corresponent. Amb la mateixa tecla es pot eliminar el punt d'aturada. Es poden afegir i treure tants punts d'aturada com es vulguin o eliminar-los tots de cop amb l'opció del menú corresponent. Polsant **F4** s'executarà el programa de forma contínua fins al primer punt d'aturada que trobi. Tornant a polsar **F4**, el programa continua executant-se fins al següent punt d'aturada que trobi, ja sigui el mateix o un altre.

Exercici 9

Recarregeu de nou el programa amb **F9**. Enfoqueu la vista del codi i situeu el cursor sobre la instrucció següent al `LD` (instrucció `ADD` situada a l'adreça 0x0014). Polseu la tecla **B** per afegir-hi un punt d'aturada, i observeu el signe **+** a l'esquerra. A continuació executeu fins al punt d'aturada polsant **F4**. Observeu l'asterisc a la dreta de l'adreça. Observeu a la vista de Registres que **R5** ha estat modificat. El seu valor és 0xffff, ja que és el primer element del vector. Si polseu **F4** repetidament, anireu executant iteració per iteració i podreu comprovar quins valors llegeix de la memòria la instrucció `LD`.

9. Modificació de les dades de memòria i registres

Per poder modificar el contingut d'un registre o posició de memòria heu d'enfocar el panell corresponent (amb **TAB**), i a continuació situar el cursor al damunt de l'element a modificar. Teniu en compte que en la vista de dades, la dada que introduïu serà de la mida igual a la que estigueu visualitzant en aquell moment. Per tant, abans de procedir a modificar, assegureu-vos que teniu la visualització adequada. A continuació, polseu **F11(o bé ? o bé ')->Modify**, o bé la tecla **M**, i introduïu l'expressió. S'accepten les mateixes expressions en llenguatge C que ja hem descrit en l'apartat anterior de guaites.

Exercici 10

Recarregueu el programa de nou (**F9**). Modifiqueu el contingut d'alguns elements de **V**:

- Per exemple, modifiqueu el primer element del vector, posant-li valor 0x0001 (visualitzant prèviament la vista de dades en mida word, en hex).
- Després, modifiqueu el darrer element del vector posant-li el valor -11. (visualitzant prèviament la vista de dades en mida word, en decimal amb signe).

A continuació executeu el programa fins al final (**F4**). Comproveu que el resultat de la suma que tenim a la variable **RES** (adreça 0x022) ha de ser ara -54.

10. Depuració de programes erronis

El programa que us donem en `s0.s` és correcte. Tanmateix, en moltes ocasions haurem programat un codi sense errors d'assemblatge però que després no fa la tasca

esperada. Ens caldrà utilitzar el depurador per trobar l'error. Però no hi ha un procediment universal de depuració ja que depèn de cada cas. El més recomanable és anar executant el programa pas a pas, o amb punts d'aturada, i comprovar que cada pas fa el que s'espera que faci, per detectar el punt on comença a fallar. Descobrir *bugs* és tot un art.

Exercici 11 (per lliurar)

Introduirem una errada en el programa i el depurarem.

- Recarregueu el programa (**F9**). Enfoqueu la vista de dades visualitzant-la en mida word i modifiqueu la posició de memòria 0x000E posant-hi el word 0x6b05.
- Executeu tot el programa fins al final (**F4**). Quin valor té **RES** (adreça 0x0022)?
- Recarregueu el programa de nou (**F9**) i executeu-lo pas a pas (**F5**). Quin comportament estrany observeu? Quin és exactament l'error que hem introduït?

OBTENIR EL PASSWORD

Enhorabona, ara ja teniu totes les eines que us calen per aturar el tren i salvar als seus passatgers. L'Orient Express destinació Vilanova acaba de sortir de l'estació, i la vostra missió és avisar al maquinista que aturi el tren però degut a un problema amb el sistema de comunicació del tren no podem donar-li l'avis.

El primer pas és saltar al darrer vagó del tren per després anar passant per tots els vagons fins a arribar al maquinista i aturar el tren. Tots els vagons tenen un mot de pass. Per obtenir la clau d'entrada al darrer vagó heu de resoldre el següent enigma:

L'SPEC Power Benchmark és un codi que mesura consum energètic de servidors amb diferents càrregues de treball (workloads) en intervals de 10% i mesura el rendiment i el consum (power).

Segons el llibre de: *Patterson, David A. Computer organization and design RISC-V edition: the hardware/software interface,* que pots trobar a la biblioteca de l'escola:

Quin es el consum mitja (averadge power) en wats d'un servidor amb arquitectura Intel Nehalem i un workload del 80%?

Pista: FIGURE 1.19 SPECpower_ssj2008 running on a dual socket 2.66 GHz Intel Xeon X5650 with 16 GB of DRAM and one 100 GB SSD disk.