Konputagailuen Egitura

Aurkibidea

1 gaia: Ordenagailuen egitura ikasgaiaren hastapena.
2 gaia: Von Neuman-en arkitektura.
3 gaia: Informazioaren irudikapen digitala: datuak: ASCII , natural-zenbakia, enteros-zenbakia
4 gaia: Eragiketa aritmetikoak eta logikoak: natura-zenbakia, enteros-zenbakia, eragiketa logikoak
5 gaia: Informazioaren irudikapen digitala: Instrukzioak.

Informazioaren irudikapen digitala: JARRAIBIDEAK

Erregistroa

- Erregistro bat zirkuitu bat da, memoria-unitate bat bezala funtzionatzen duena eta datu bakar bat edo makina-instrukzio bat gordetzen duena.
- Erregistroak:
  - bit-sekuentzia batez osatutako hitz bat 'gordetzen' dute.
  - dimentsio bateko gelaxkak dira, eta gelaxka bakoitzak bit bat gordetzen du.
- Bere tamaina, normalean, 8 byteko multiplo bat da, eta izen bat jasotzen du erreferentzia izan ahal izateko, adibidez, RAX
  - 8 bit: 1 Byte
  - 16 bit: Word. Arrazoi historikoengatik. (gogoratu hitz baten tamaina beste testuinguru batean dagokion makinaren araberakoa dela)

*. **32 bit: double word** 64 bit: quad word * Gelaxkak zerotik hasita zerrendatzen dira. * LSB: Least Significant Bit bit pisu txikiena * MSB: Most Significant Bit pisu handieneko bit da

+

MSB LSB

7	6	5	4	3	2	1	0

RTL lengoaia: transferentzia-operadorea

- RTL lengoaia lengoaia bat da MAKINA JARRAIBIDEAK deskribatzeko: Register Transfer Language (RTL)
- RTL lengoaiaren helburua da PUZak exekutatzen dituen makina-jarraibideak adierazi ahal izatea, hala nola batu (ADD), kendu (SUB), mugitu (MOV), etab. Deskribapena PUZren barne-erregistroen arteko datu-transferentziaren mailan edo barne-erregistroen eta kanpo-memoriaren artean egiten da.
  - Transferentzia* eragiketa gezi batekin irudikatzen da, eskuinetik ezkerrera
  - Transferentzia-operadorea < -
  - Transferentziaren epaia: R2 < -R1

- R1i iturburu-erregistroa deitzen zaio, eta R2RI xede-erregistroa
- Interpretazioa: R1 erregistroaren edukia kopiatu edo transferitzen dugu R2 erregistroan

RTL lengoaia: beste sententzia batzuk

Baldintzapeko epaia:
If (K1 = 1) then R2 ←R1
K1: R2 ←R1
Transferentzia edo kopia K1 egia bada bakarrik egiten da, hau da, K1ek 1 balio logikoa (TRUE) balio badu.

Aldi bereko epaia:
Operadore koma
K3: R2 ←R1, R3 ←R1
K3 egia bada, R1 edukia R2 eta R3 kopietan kopiatuko da ---

Jarraibideen sintaxia INTEL lengoaian

- Jarraibideen formatuari, mihiztatze-lengoaian, jarraibideen "sintaxi" esaten zaio.
- ASM SINTAXIA: Lan-Eragiketaren Kodea-Eragiketa-Eragiketa-Eragiketa-Iruzkina-
- Arkitekturak x86-64 eta x86

Table 1. Sintel Axia: Egitura

label:	op_mnemonic	operand_destination	,	operand_source	#comment

Jarraibideen sintaxia INTEL lengoaian: Adibidea

- Adibidea:

```
begizta: sub rsp, 16; RSP < - RSP-16. Kenketa
         je bucle; je: jump equal:
                 ; jauzia, azken eragiketak zero emaitza izan bazuen
batura: add eax, esi; EAX < - EAX+M [ESI]. Batu
        mov ax, [emaitza]; AX < - M [emaitza].
        ; AX erregistroan kopiatu emaitzaren memoria-posizioaren edukia


emaitza: "memoria-erreserba"
```

Jarraibideen sintaxia AT&T telefono-konpainiaren hizkuntzan

- ASM SINTAXIA: Lan-Eragiketaren Kodea-Eragiketa-Eragiketa-Eragiketa-Iruzkina-
- Arkitekturak x86-64 eta x86

Table 2. AT&T sintaxia: Egitura

label:	op_mnemonic	operand_source	,	operand_destination	#comment

- Intelen lengoaiarekiko alde handia bi operandoen ordena da
- Beste alde txiki bat operatuen aurrizkiak dira, eragiketa bideratzeko modua adierazteko

Jarraibideen sintaxia AT & T lengoaian: Adibidea

- Adibidea:

```
begizta: sub $16, %rsp; RSP < - RSP-16. Kenketa
         je begizta ; je: jump equal:
                    ; jauzia, azken eragiketak zero emaitza izan bazuen
batura:  add %esi, %eax; EAX < - EAX+M [ESI]. Batu
         mov emaitza, %ax; AX < - M [emaitza] ; AX erregistroan kopiatu emaitzaren
memoria-posizioaren edukia

emaitza: "memoria-erreserba"
```

AT&T sintaxia

Mihiztatze-lengoaiaren sintaxia mihiztatze-prozesuaren (**assembler**) "itzultzailearen" araberakoa da; kasu honetan, GAS assemblerra erabiltzen da.

```
ETIKETA: Lehenengo zutabean zehazten da. Atzizkia du ":"
OPERAZIO-KODEA: Eragiketa intuitiboki interpretatzen laguntzen duten sinbolo
mnemonikoak erabiltzen dira.
     Adibidez: ADD batu, MOV mugitu, SUB kenketa,...
ITURRIA ETA/EDO HELMUGA ERAGINGAIA:
 datu alfanumerikoa: irudikapen alfanumerikoa → 16
 berehalako helbideratzea: "$" aurrizkia
 eremu-memoriaren helbidea: etiketa → emaitza
 helbideratze zuzena
 PUZren barne-erregistroak: %rax, %rbx, %rsp, %esi,..
        "%" aurrizkiak esan nahi du izenak erregistro bati egiten diola erreferentzia
 datu-tamaina lanean: mnemonikoen atzizkiak:
         q (quad): 8 byte, l (long): 4 byte, w (word): 2 byte, b (byte): 1 byte.
 Atzizkirik gabe, aipatutako erregistroaren tamainaren muga hartzen da
 eta mugarik ez badago, itzultzaileak akatsaren berri emango du.
```

Eragingaiak: Helbideratzeko modua

- Helbideratzeak:

  BEREHALAKOA: Eragiketaren balioa instrukzioaren eragiketa-kodearen ondoren dago. Iturriaren eragiketa bakarrik zehazten da.
  sintaxia: operandoaren balioa **$** aurrizkiarekin adierazten da.
  adibidez: **movl $0xabcd1234, %ebx**. Iturri-eragiketa 0xABCD1234 balioa da

  ERREGISTROA: Operandoaren balioa PUZko erregistro batean dago.
  sintaxia: Erregistroaren izena* %* aurrizkiarekin.
  adibidez: **movl %eax, %ebx**. Iturria operatzen duena EAX ERREGISTROA da eta helmuga EBX

ERREGISTROA da

ZUZENA: Memoria Nagusian biltegiratutako eragiketaren helbide efektiboa eragiketaren eremuan zehaztutako etiketak adierazitako helbide absolutua da. Programatzaileak zuzeneko helbideratzea erabiltzen du, baina konpiladoreak programa-kontagailuari dagokion helbideratze bihurtzen du. Ikusi noranzkoa desplazamenduekin.
sintaxia: programatzaileak definitutako etiketa
adibidez: **je somePlace**. Jauzi egin somePlace etiketak markatutako helbidera, aurreko eragiketaren emaitzak RFLAG erregistroko ZF = 1 flag aktibatzen badu.

INDEXATZEA: Operandoaren balioa memorian dago. Memoriara zuzendutako benetako helbidea MAS SCALE erregistro-balioaren BATURA da, GEHI offseta. 'EA = Offset+R_Base+R_aurkibidea*Scale'
sintaxia: koma bidez eta parentesi artean bereizitako balioen zerrenda (base_register, index_register, scale) eta offset bat aurretik duela.
adibidez: **mov $0x6789cdef, -16(%edx, %eax, 4)**. Helmugaren benetako helbidea **EDX + EAX*4 - 16** da.

Eragingaiak: Helbideratzeko modua

- Helbideratzeak:

  ZEHARKAKOA: Indexazio modu orokorra + (base_register) + atalean zehazten badugu, eragingaiaren helbidea ez da indexazio baten bidez lortzen, baizik eta norabide eraginkorra rdx edukia da eta, beraz, zeharkako eragiketan sartzen da.
  sintaxia: (base_register)
  adibidez: **mov $0x6789cdef, (%edx)**. Helmugaren benetako helbidea EDX da. EDX punta-puntakoa da.

  ERLATIBOA: oinarrizko erregistroa gehi offset bat: Eragingaiaren balioa memorian dago. Eragiketaren benetako helbidea oinarrizko erregistro batean jasotako balioaren eta offset balio baten arteko batura da.
  sintaxia: parentesi eta offset arteko erregistroa parentesia baino lehen.
  adibidez: **mov $0xaabbccdd, -12(%eax)**. Helmugako operazioaren benetako helbidea EAX-12 da.

Eragiketak bideratzeko moduak: Adibideak

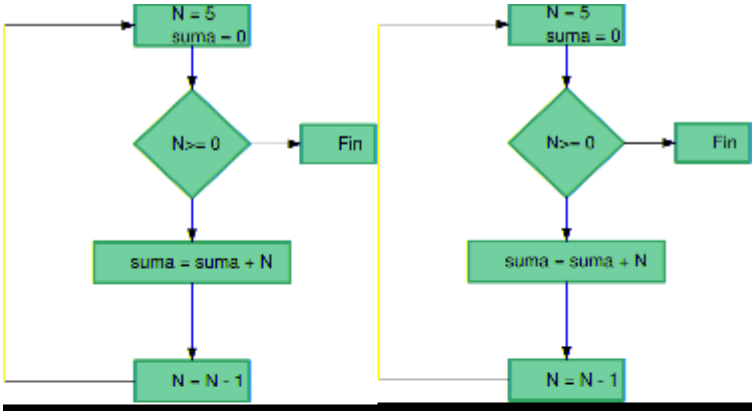
Table 3. Eragingaiak helbideratzeko moduak

Eragingai helbideratzea	Eragingai balioa	Motako izena
`$0`	`Zero Balioa`	`Berehalakoa`
`%rax`	`RAX`	`Erregistroa`
`loop_exit`	`M [loop_exit]`	`Zuzena`
`data_items (, %rdi, 4)`	`M [data_item + 4*RDI]`	`Indexatua`
`( %rbx)`	`M [RBX]`	`Zeharkakoa`
`( %rbx, %rdi, 4)`	`M [RBX + 4*RDI]`	`Zeharkakoa Indexatua`

- M [loop_exit]: zuzena, loop_exit kanpo-memoriaren helbidea baita, eta M-k kanpo-memoria adierazten baitu.
- M [RBX]: zeharkakoa; izan ere, RBX barne-memoriaren helbide bat da, eta M-k kanpo-memoria adierazten du: Kanpo-memoriara barne-memoriatik sartzen da.

Sum1toN programa: Organigrama

- Kalkulatu lehenengo 5 zenbaki arrunten batura



Sum1toN programa C hizkuntzan

```
/*
Egitaraua: sum1toN.c
Deskribapena: 1,2,3 seriearen batuketa egiten du,... N

Prozesadorearen arkitektura: Algoritmo honen programazioa C lengoaian EZ DAGO
pozesatzailearen arkitekturaren menpe. Zergatik eta Nola da posible?
Hizkuntza: C99
Deskribapena: Lehenengo 5 zenbaki arrunten batura
Sarrera: Aldagai batean zehaztuta
Irteera: Irteerarik gabe
Konpilazioa: gcc -m32 -g -o sum1toN sum1toN.c -> -g: modulu bitar araztua
- > -m32: arkitektura modulu bitarra x86-32 bit
S.O: GNU/LINUX 4.10 ubuntu 17.04 x86-64
Liburutegia:/usr/lib/x86_64-linux-gnu/libc.so
PUZ: Intel (R) Core (TM) i5-6300U CPU @ 3.0GHz
Konpilatzailea: gcc version 6.3
Mihiztatzailea: GNU assembler 2.28 bertsioa
Estekaketa(Linker/Loader): GNU ld (GNU Binutils for Ubuntu) 2.28
Irakasgaia: Konputagailuen egitura
Data: 2023/09/17
Egilea: Candido Aramburu
* /

#include < stdio.h>// printf funtzioaren liburutegiaren goiburua ()

// programan sartzeko funtzioa
void main (void)
```

```
{
// Tokiko aldagaien deklarazioa
  char suma = 0;
  char n = 0b101;
// begizta
  while (n>0) {
    suma+ = n;
   n--;
}
  printf ("\n Batura da = %d\n", batura);
}
```

Sum1toN programa x86 makinarako, AT & T lengoaian

- ATT lengoaia mihiztatzailea arkitekturarako x86-32

```
# # #Programa: sum1toN.s
# # #Deskribapena: 1,2,3 serie,... N
# #Prozesadorearen arkitektura: x86 32 bit
# #gcc -m32 -g -nostartfiles -o sum1toN sum1toN.s
# # #Mihiztatzea --32 --gstabs fuente.s o objeto.o
# # #linker -> ld -melf_i386 -I/lib/i386-linux-gnu/ld-linux.so.2 edo objektua-lc
exekutagarria

# #Aldagaien aitorpena
    .section .data

n: .int 5

.globala _start

##Kodearen hasiera
    .section .text
_start:
    mov $0,%ecx #ECXk batura aldagaia inplementatzen du
    mov n,%edx
begizta:
    add %edx,%ecx
    $1, %edx
    jnz begizta

    mov %ecx,%ebx #el argumento de salida al S.O. a través de EBX según convenio

# #irteera
    mov $1, %eax #sistema eragilerako deiaren kodea:
    int $0x80 #sistema eragilerako deia


.end
```

Sum1toN programa x86 makinarako, Intel lengoaian

- INTEL eta assembler nasm mihiztatzeko lengoaia

```
;;; Programa: sum1toN.asm
;;; Deskribapena: 1,2,3 seriearen batura egiten du,... N
;;; Prozesadorearen arkitektura: x86 32 bit
;;; INTEL hizkuntza
;;; Assembler NASM

;;; nasm -hf -> f aukeraren laguntza
;;; Nasm mihiztadura -g -f elf sum1toN.asm -o sum1toN.o
;; linker -> ld -m elf_i386 -o sum1toN sum1toN.o

BITS 32; CPU MODE
;;; Aldagaien aitorpena
    section .data

n: dd 5; 4 byte

globala _start

;; Kodearen hasiera
    section .text
_start:
  mov ecx, 0; ECXk batura aldagaia inplementatzen du
  mov edx, [n]; n aldagaiaren ezizena da EDX
  begizta:
  add ecx, edx
  sub edx, 1
  jnz begizta

  mov ebx, ecx ; S.O.ra EBXren bidez irteteko argudioa hitzarmenaren arabera

;; irteera
  mov eax, 1 ; sistema eragilerako deiaren kodea:
  int 0x80 ; sistema eragilerako deia
```