75.03 & 95.57 Organización del Computador

U4 – LENGUAJE ENSAMBLADOR

U4 – Lenguaje ensamblador

- Introducción
  - Lenguaje de máquina / código máquina
    - Definición: “Representación binaria de un programa de computadora el cual es leído e interpretado por el computador. Consiste en una secuencia de instrucciones de máquina”
  - Lenguaje ensamblador
    - Definición: “Representación simbólica del lenguaje de máquina de un procesador específico.”

U4 – Lenguaje ensamblador

- Lenguaje ensamblador
  - Transición entre lenguaje de máquina y ensamblador.

Address	Contents			
101	0010	0010	101	2201
102	0001	0010	102	1202
103	0001	0010	103	1203
104	0011	0010	104	3204
201	0000	0000	201	0002
202	0000	0000	202	0003
203	0000	0000	203	0004
204	0000	0000	204	0000

(a) Binary program

Address	Contents
101	2201
102	1202
103	1203
104	3204
201	0002
202	0003
203	0004
204	0000

(b) Hexadecimal program

Address	Instruction	
101	LDA	201
102	ADD	202
103	ADD	203
104	STA	204
201	DAT	2
202	DAT	3
203	DAT	4
204	DAT	0

(c) Symbolic program

Label	Operation	Operand
FORMUL	LDA	I
	ADD	J
	ADD	K
	STA	N
I	DATA	2
J	DATA	3
K	DATA	4
N	DATA	0

(d) Assembly program

Figure 13.13 Computation of the Formula N = I + J + K

U4 – Lenguaje ensamblador

- Lenguaje ensamblador
  - ¿Por qué usarlo aún hoy?
    - Debugging y verificación
    - Desarrollar compiladores
    - Sistemas embebidos
    - Drivers de hardware y código de sistema
    - Acceder a instrucciones no disponibles en un lenguaje de alto nivel
    - Código automodificable
    - Optimizar código en tamaño
    - Optimizar código en velocidad
    - Biblioteca de funciones

U4 – Lenguaje ensamblador

- Lenguaje ensamblador
  - Elementos que lo componen
    - Etiquetas
    - Mnemónicos
    - Operandos
    - Comentarios

U4 – Lenguaje ensamblador

- Lenguaje ensamblador
  - Tipos de sentencias
    - Instrucciones
    - Directivas (pseudoinstrucciones)
    - Macroinstrucciones

U4 – Lenguaje ensamblador

- Lenguaje ensamblador
  - Ejemplo Intel x86

```
global          _main
section         .data
      num1  dd   5
      num2  dd   6
section         .bss
      resMay    resd  1
      resMen    resd  1
section         .text
_main:
      mov       eax,[num1]
      mov       ebx,[num2]
      add       eax,ebx
      cmp       eax,10

      jg        mayor
      mov       [resMen],eax

      jmp       fin
mayor:
      mov       [resMay],eax
fin:
      ret
```

U4 – Lenguaje ensamblador

- Lenguaje ensamblador
  - Ejemplo ARM

```
        AREA    ARMex, CODE, READONLY
                                ; Name this block of code ARMex
        ENTRY                   ; Mark first instruction to execute
start
        MOV     r0, #10         ; Set up parameters
        MOV     r1, #3
        ADD     r0, r0, r1      ; r0 = r0 + r1
stop
        MOV     r0, #0x18       ; angel_SWIreason_ReportException
        LDR     r1, =0x20026    ; ADP_Stopped_ApplicationExit
        SVC     #0x123456       ; ARM semihosting (formerly SWI)
        END                     ; Mark end of file
```

U4 – Lenguaje ensamblador

- Lenguaje ensamblador
  - Ejemplo IBM Mainframe

```
*********************************************************************
* Copyright 2005 Automated Software Tools Corporation               *
* This source code is part of z390 assembler/emulator package       *
* The z390 package is distributed under GNU general public license  *
* Author - Don Higgins                                              *
* Date   - 09/30/05                                                 *
*********************************************************************
TESTDCBB SUBENTRY
         WTO   'TESTDCBB TEST USE OF DCBREC WITH READ/WRITE'
         OPEN  (SYSUT1,(INPUT),SYSUT2,(OUTPUT),SYSOUT,(OUTPUT))
LOOP     EQU   *
         READ  DECB1,SF,SYSUT1
         CHECK DECB1
         AP    PTOT,=P'1'
         MVC   DTOT,=X'40202020'
         ED    DTOT,PTOT
         PUT   SYSOUT,MSG
         WRITE DECB2,SF,SYSUT2
         CHECK DECB2
         B     LOOP
EOF      CLOSE (SYSUT1,,SYSUT2,,SYSOUT)
         WTO   'TESTDCBB ENDED OK'
         SUBEXIT
SYSUT1   DCB   DSORG=PS,DDNAME=SYSUT1,EODAD=EOF,MACRF=R,               X
               RECFM=F,BLKSIZE=80,RECORD=RECORD
SYSUT2   DCB   DSORG=PS,DDNAME=SYSUT2,MACRF=W,RECFM=F,BLKSIZE=80,      X
               RECORD=RECORD
SYSOUT   DCB   DSORG=PS,DDNAME=SYSOUT,RECFM=FT,BLKSIZE=120,MACRF=PM
PTOT     DC    PL2'0'
MSG      DS    0CL120
         DC    C'REC#='
DTOT     DC    CL4' ',C' TEXT='
RECORD   DC    CL80' '
         DC    (MSG+120-*)C' '
         DCBD
         DECBD
         EQUREGS
         END
```

U4 – Lenguaje ensamblador

- Traducción versus Interpretación
  - Traductor
    - Definición: “Programa que convierte un programa de usuario escrito en un lenguaje (fuente) en otro lenguaje (destino)
    - Clasificación:
      - Compiladores
      - Ensambladores
  - Intérprete
    - Definición: “Programa que ejecuta directamente un programa de usuario escrito en un lenguaje fuente”

U4 – Lenguaje ensamblador

- Traducción versus Interpretación
  - Traducción
    - Lenguajes compilados:
      - C, C++, Go, Rust
    - Lenguajes ensambladores:
      - Intel 64/IA-32, ARM, SPARC, MIPS, IA-64 (Itanium)
  - Interpretación
    - Lenguajes interpretados:
      - Python, JavaScript, Ruby
    - Bytecode
      - Java

U4 – Lenguaje ensamblador

- Ensambladores
  - Definición: “Programa que traduce un programa escrito en lenguaje ensamblador y produce código objeto como salida”
  - Traducción 1 a 1 a lenguaje máquina
  - Hay dos tipos:
    - Dos pasadas
    - Una pasada

U4 – Lenguaje ensamblador

- Ensambladores
  - Dos pasadas
    - Primera
      - Definición de etiquetas → Tabla de símbolos
      - LC: location counter (empieza en 0 con el 1er byte del código objeto ensamblado)
      - Se examina cada sentencia de lenguaje ensamblador
      - Determina la longitud de la instrucción de máquina (reconoce opcode + modo de direccionamiento + operandos) para actualizar el LC
      - Revisa directivas al ensamblador.
        - Ejemplo: definición de áreas de storage
      - Por cada etiqueta encontrada se fija si está en la tabla de símbolos. Si no lo está la agrega (si es la definición, registra el LC como tal, sino lo registra como referenciando a la etiqueta)

U4 – Lenguaje ensamblador

- Ensambladores
  - Dos pasadas
    - Segunda (Traducción)
      - Traduce el mnemónico en el opcode binario correspondiente
      - Usa el opcode para determinar el formato de la instrucción y la posición y tamaño de cada uno de los campos de la instrucción
      - Traduce cada nombre de operando en el registro o código de memoria apropiado
      - Traduce cada valor inmediato en un string binario en la instrucción
      - Traduce las referencias a etiquetas en el valor apropiado de LC usando la tabla de símbolos
      - Setear otros bits necesarios en la codificación de la instrucción.
        - Ejemplo: indicadores de modo de direccionamiento, bits de código de condición, etc.

U4 – Lenguaje ensamblador

- Referencias
  - “Computer Organization and Architecture – Designing for Perfomance” 9na edición. William Stallings (http://williamstallings.com/ComputerOrganization/)
  - “Structured Computer Organization” 6ta edición. Andrew Tanenbaum / Todd Austin (http://www.pearsonhighered.com/educator/product/Structured-Computer-Organization-6E/9780132916523.page)