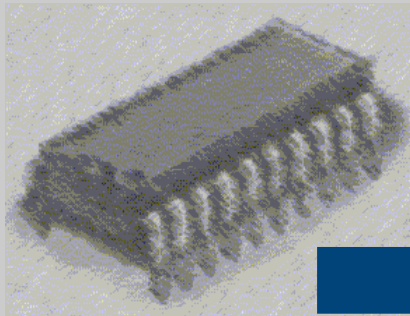


*Trabajo evaluable 1*

## **Comparativas del número de ciclos de ejecución de un programa en ensamblador**

***Arquitectura de  
Computadores I***



I. T. Informática de Sistemas

Curso 2009-2010

## Trabajo evaluable 1

### Opción 1

Sea el fragmento de programa escrito en ensamblador del i8086 siguiente:

```
CLD
LEA    SI, BUFFER1
LEA    DI, BUFFER2
MOV    CX, 100
REP    MOVSB
```

Dicho fragmento copia la cadena `buffer1` en la cadena `buffer2`, ambas de la misma longitud.

#### Se pide:

- Completar el fragmento anterior con las definiciones necesarias en el `.data` y completar el programa en el `.code`
  - Realizar el mismo programa pero sin emplear instrucciones de manejo de cadenas
  - Calcular para ambos programas el número de ciclos de instrucción necesarios para completarlos (consultar el apéndice en este mismo documento)
  - Indicación de tiempos que se ha tardado en realizar el trabajo. Se deben indicar los tiempos (tanto de estudio en casa, en la biblioteca, consultas en internet, consultas en tutorías) empleados en: encontrar la solución, en redactar el trabajo y en crear la memoria.
- 
-

## Opción 2

Sea el fragmento de programa escrito en ensamblador del i8086 siguiente:

```
MOV     AX, DS
MOV     ES, AX
CLD
MOV     AL, 'A'
MOV     CX, 100
LEA     DI, Buffer1
REP     STOSB
```

Dicho fragmento **almacena** el carácter A en la cadena `buffer1`.

### Se pide:

- Completar el fragmento anterior con las definiciones necesarias en el `.data` y completar el programa en el `.code`
  - Realizar el mismo programa pero sin emplear instrucciones de manejo de cadenas
  - Calcular para ambos programas el número de ciclos de instrucción necesarios para completarlos (consultar el apéndice en este mismo documento)
  - Indicación de tiempos que se ha tardado en realizar el trabajo. Se deben indicar los tiempos (tanto de estudio en casa, en la biblioteca, consultas en internet, consultas en tutorías) empleados en: encontrar la solución, en redactar el trabajo y en crear la memoria.
- 
-

## Apéndice sobre el cálculo de ciclos de las instrucciones

Cálculo de la dirección efectiva en función del modo de direccionamiento	Ciclos en el i8088	Ejemplo
Directo	6	MOV AX, ADDR
Relativo a registro BX, BP, SI, DI	5	MOV AX, [BP]
Desplazamiento más registro base o registro índice BX + Disp, BP + Disp SI + Disp, DI + Disp	9	MOV AX, ADDR[BP]
Registro base + índice		
BP + DI, BX + SI	7	MOV AX, [BP+DI]
BP + SI, BX + DI	8	MOV AX, [BX+DI]
Desplazamiento más registri base + registro índice		
BP + DI + Disp BX + SI + Disp	11	MOV AX, ADDR[BP+DI]
BP + SI + Disp BX + DI + Disp	12	MOV AX, ADDR[BP+SI]

### Notas:

- Añadir 2 ciclos de reloj si se sobrescriben segmentos
- Cada referencia a memoria requiere 4 ciclos adicionales

### Ciclos empleados por las instrucciones más usadas y de cadena.

Dónde pone EA se refiere a los cálculos necesarios para calcular los ciclos de la tabla superior

### ADD

Operands	Clocks Byte (word)	Transfers	Bytes	Example
register, register	3	-	2	ADD BX, CX
accumulator, immediate	4	-	2-3	ADD AX, 256
register, immediate	4	-	3-4	ADD BL, 4
register, memory DI, [DX]	9 (13) + EA	1	2-4	ADD
memory, register	16 (24) + EA	2	2-4	ADD
TOTAL, BL				
memory, immediate	17 (25) + EA	2	3-6	ADD
RESULT, 3				

## Diseño de un repertorio de instrucciones

## SUB

Operands	Clocks byte (word)	Transfers	Bytes	Example
register, register	3	-	2	SUB DX, BX
register, memory DX, TOTAL	9 (13) + EA	1	2-4	SUB
memory, register RATE, CL	16 (24) + EA	2	2-4	SUB
accumulator, immediate	4	-	2-3	SUB AH, 25
register, immediate DX, 5280	4	-	3-4	SUB
memory, immediate RES, 1032	17 (25) + EA	2	3-6	SUB

## MUL

Operands	Clocks	Transfers	Bytes	Example
reg8	70-77	-	2	MUL CH
reg16	118-133	-	2	MUL BX
mem8	(76-83) + EA	1	2-4	MUL A_BYTE
mem16	(128-143) + EA	1	2-4	MUL A_WORD

## IMUL

Operands	Clocks	Transfers	Bytes	Example
reg8	80-98	-	2	IMUL CL
reg16	128-154	-	2	IMUL BX
mem8	(86-104) + EA	1	2-4	IMUL BITE
mem16 WORD [BP] [DI]	(138-164) + EA	1	2-4	IMUL

## DIV

Operands	Clocks	Transfers	Bytes	Example
reg8	0-90	-	2	DIV BL
reg16	4-162	-	2	DIV BX
mem8	(86-96) + EA	1	2-4	DIV VYUP
mem16 NCONQUER [SI]	(154-172) + EA	1	2-4	DIV

## IDIV

Operands	Clocks	Transfers	Bytes	Example
reg8	101-112	-	2	IDIV CL
reg16	165-184	-	2	IDIV DX
mem8 BYTE [SI]	(107-118) + EA	1	2-4	IDIV
mem16 [BX].WORD_ARRAY	(175-194) + EA	1	2-4	IDIV

## INC

Operands	Clocks byte (word)	Transfers	Bytes	Example
reg16	2	-	1	INC BX
reg8	3	-	2	INC BL
memory THETA [BX]	15 (23) + EA	2	2-4	INC

## Diseño de un repertorio de instrucciones

## DEC

Operands	Clocks	Transfers	Bytes	Example
reg16	2	-	1	DEC BX
reg8	3	-	2	DEC BL
memory MATRIX[SI]	15 (23) + EA	2	2-4	DEC

## MOV

Operands	Clocks byte (word)	Transfers	Bytes	Example
register, register	2	-	2	MOV BX,CX
memory, accumulator MEM_DEST,AL	10 (14)	1	3	MOV
accumulator, memory AX, MEM_SOURCE	10 (14)	1	3	MOV
memory, register MEM_DEST,BX	9 (13) + EA	1	2-4	MOV
register, memory BX, MEM_SOURCE	8 (12) + EA	1	2-4	MOV
register, immediate BX, 0F6CDh	4	-	2-3	MOV
memory, immediate MASK, 0F6CDh	10 (14) + EA	1	3-6	MOV
seg-reg, reg16	2	-	2	MOV DS, BX
seg-reg, mem16 DS, SEGMENT_VAL	8 (12) + EA	1	2-4	MOV
reg16, seg-reg	2	-	2	MOV BP, SS
memory, seg-reg CODE_VAR, CS	9 (13) + EA	1	2-4	MOV

## PUSH

Operands	Clocks	Transfers	Bytes	Example
register	15	1	1	PUSH BX
seg-reg (CS illegal)	14	1	1	PUSH ES
memory PARAMETERS	24 + EA	2	2-4	PUSH

## POP

Operands	Clocks	Transfers	Bytes	Example
register	12	1	1	POP CX
seg-reg (CS illegal)	12	1	1	POP ES
memory	25 + EA	2	2-4	POP VALUE

## LEA

Operands	Clocks	Transfers	Bytes	Example
reg16, mem16 BX, MEM_ADDR	2 + EA	-	2-4	LEA

## NEG

Operands	Clocks byte (word)	Transfers	Bytes	Example
register	3	-	2	NEG DL
memory COEFFICIENT	16 (24) + EA	2	2-4	NEG

## Diseño de un repertorio de instrucciones

**JZ/JE**

<b>Operands</b>	<b>Clocks</b>	<b>Transfers</b>	<b>Bytes</b>	<b>Example</b>
short-label	16 or 4	-	2	JE ZERO

**JNZ/JNE**

<b>Operands</b>	<b>Clocks</b>	<b>Transfers</b>	<b>Bytes</b>	<b>Example</b>
short-label NOT_EQUAL	16 or 4	-	2	JNE

**JG**

<b>Operands</b>	<b>Clocks</b>	<b>Transfers</b>	<b>Bytes</b>	<b>Example</b>
short-label	16 or 4	-	2	JG GREATER

**JL**

<b>Operands</b>	<b>Clocks</b>	<b>Transfers</b>	<b>Bytes</b>	<b>Example</b>
short-label	16 or 4	-	2	JG GREATER

**JLE**

<b>Operands</b>	<b>Clocks</b>	<b>Transfers</b>	<b>Bytes</b>	<b>Example</b>
short-label	16 or 4	-	2	JG GREATER

**JMP**

<b>Operands</b>	<b>Clocks</b>	<b>Transfers</b>	<b>Bytes</b>	<b>Example</b>
short-label	15	-	2	JMP
ROPE_NEAR near-label	15	-	3	JMP
SAME_SEGMENT far-label	15	-	5	JMP
FAR_LABEL memptr16	18 + EA	-	2-4	JMP
SAME_SEG regptr16	11	-	2	JMP BX
memptr32 NEXT_SEG	24 + EA	-	2-4	JMP

**CLD/STD**

<b>Operands</b>	<b>Clocks</b>	<b>Transfers</b>	<b>Bytes</b>	<b>Example</b>
no operands	2	-	1	CLD

**LOOP**

<b>Operands</b>	<b>Clocks</b>	<b>Transfers</b>	<b>Bytes</b>	<b>Example</b>
short-label	17/5	-	2	LOOP AGAIN

**LOOPE / LOOPZ**

<b>Operands</b>	<b>Clocks</b>	<b>Transfers</b>	<b>Bytes</b>	<b>Example</b>
short-label AGAIN	18 or 6	-	2	LOOPE

**LOOPNE / LOOPNZ**

<b>Operands</b>	<b>Clocks</b>	<b>Transfers</b>	<b>Bytes</b>	<b>Example</b>
short-label AGAIN	19 or 5	-	2	LOOPNE

## Diseño de un repertorio de instrucciones

**OR**

<b>Operands</b>	<b>Clocks byte (word)</b>	<b>Transfers</b>	<b>Bytes</b>	<b>Example</b>
register, register	3	-	2	OR CH, DL
register, memory	9 (13) + EA	1	2-4	OR BX,
MEM_OR_Y				
memory, register	16 (24) + EA	2	2-4	OR
MEM_OR_Y, BX				
accumulator, immediate	4	-	2-3	OR
AL, 01110110b				
register, immediate	4	-	3-4	OR
CX, 00FFh				
memory, immediate	17 (25) + EA	2	3-6	OR
MEM_WORD, 76h				

**AND**

<b>Operands</b>	<b>Clocks byte (word)</b>	<b>Transfers</b>	<b>Bytes</b>	<b>Example</b>
register, register	3	-	2	AND AL, DL
register, immediate	4	-	3-4	AND
CX, 0FFh				
accumulator, immediate	4	-	2-3	AND
AX, 01000010b				
register, memory	9 (13) + EA	1	2-4	AND
CX, MASK				
memory, register	16 (24) + EA	2	2-4	AND
VALUE, BL				
memory, immediate	17 (25) + EA	2	3-6	AND
STATUS, 01h				

**XOR**

<b>Operands</b>	<b>Clocks</b>	<b>Transfers</b>	<b>Bytes</b>	<b>Example</b>
register, register	3	-	2	XOR CX, BX
register, memory	9 (13) + EA	1	2-4	XOR
CL, MASK_BYTE				
memory, register	16 (24) + EA	2	2-4	XOR
ALPHA[SI], DX				
accumulator, immediate	4	-	2-3	XOR
AL, 01000001b				
register, immediate	4	-	3-4	XOR
SI, 00C2h				
memory, immediate	17 (25) + EA	2	3-6	XOR
RETURN_CODE, 0D2h				

**NOT**

<b>Operands</b>	<b>Clocks byte (word)</b>	<b>Transfers</b>	<b>Bytes</b>	<b>Example</b>
register	3	-	2	NOT DX
memory	16 (24) + EA	2	2-4	NOT MASK



## Diseño de un repertorio de instrucciones

## SHL / SHR / SAL / SAR / ROL / ROR / RCL / RCR

Operands	Clocks byte (word)	Transfers	Bytes	Example
register, 1	2	-	2	SHL AL, 1
register, CL	8 + 4/bit	-	2	SHL SI, CL
memory, 1	15 (23) + EA	2	2-4	SHL WORD, 1
memory, CL	20 (28)+EA+4/bit	2	2-4	SHL BYTE, CL

## TEST

Operands	Clocks byte (word)	Transfers	Bytes	Example
register, register	3	-	2	TEST SI, DX
register, memory SI, MASK	9 (13) + EA	1	2-4	TEST
accumulator, immediate AL, 00000100b	4	-	2-3	TEST
register, immediate CX, 1234	5	-	3-4	TEST
memory, immediate PARAM, 1F1Fh	11 + EA	-	3-6	TEST

## CMP

Operands	Clocks byte (word)	Transfers	Bytes	Example
register, register	3	-	2	CMP CX, BX
register, immediate	4	-	3-4	CMP BL, 02h
accumulator, immediate AL, 00010110b	4	-	2-3	CMP
register, memory DH, ALPHA_BETA	9 (13) + EA	1	2-4	CMP
memory, register TOTAL, SI	9 (13) + EA	1	2-4	CMP
memory, immediate VALUES, 3420h	10 (14) + EA	1	3-6	CMP

## CALL

Operands	Clocks byte (word)	Transfers	Bytes	Example
near-proc NEAR_PROC	19 (23)	1	3	CALL
far-proc FAR_PROC	28 (36)	2	5	CALL
memptr 16 PROC_TABLE[SI]	21 (29) + EA	2	2-4	CALL
regptr 16	16 (24)	1	2	CALL AX
memptr 32 [BX].ROUTINE	37 (57) + EA	4	2-4	CALL

## RET

Operands	Clocks	Transfers	Bytes	Example
(intrasegment, no pop)	20	1	1	RET
(intrasegment, with pop)	24	1	3	RET 4
(intersegment, no pop)	32	2	1	RET
(intersegment, pop)	31	2	3	RET 2

## Diseño de un repertorio de instrucciones

## INT

Operands	Clocks byte (word)	Transfers	Bytes	Example
immed8 (type = 3)	52	5	1	INT 3
immed8 (type <> 3)	51	5	2	INT 21

## IRET

Operands	Clocks	Transfers	Bytes	Example
no operands	32	3	1	IRET

## IN

Operands	Clocks byte (word)	Transfers	Bytes	Example
accumulator, immed8	10 (14)	1	2	IN AL, 45h
accumulator, DX	8 (12)	1	1	IN AX, DX

## OUT

Operands	Clocks byte (word)	Transfers	Bytes	Example
immed8, accumulator	10 (14)	1	2	OUT 254, AX
DX, accumulator	8 (12)	1	1	OUT DX, AL

## CMPS / CMPSB / CMPSW

Operands	Clocks byte (word)	Transfers	Bytes	Example
dest, source STR1, STR2	22 (30)	2	1	CMPS
(repeat) dest, source STR1, STR2	9 + 22 (30) / rep	2 / rep	1	REPE CMPS

## LODS / LODSB / LODSW

Operands	Clocks byte (word)	Transfers	Bytes	Example
source-str	12 (16)	-	1	LODS LIST
(repeat) source-str LIST	9+13 (17) / rep	1 / rep	1	REP LODS

## MOVS / MOVSB / MOVSW

Operands	Clocks byte (word)	Transfers	Bytes	Example
dest, source WORD_BUFF, INPUT	18 (26)	2	1	MOVS
(repeat) dest, source	9+17 (25) / rep	2 / rep	1	REP MOVSW

## STOS / STOSB / STOSW

Operands	Clocks byte (word)	Transfers	Bytes	Example
dest-string WORD_ARRAY	11 (15)	1	1	STOS
(repeat) dest-string BYTE_ARRAY	9 + 10 (14) / rep	1 / rep	1	REP STOS

## Diseño de un repertorio de instrucciones

## SCAS / SCASB / SCASW

Operands	Clocks byte (word)	Transfers	Bytes	Example
dest-str	15 (19)	1	1	SCAS
WORD_TABLE (repeat) dest-str	9 + 15 (19)/rep	1/rep	1	REPNE SCAS
BYTE_TABLE				

## REP / REPE / REPZ / REPNE / REPNZ

Operands	Clocks	Transfers	Bytes	Example
-	2	-	1	REP MOVS
TO, FROM				