Universitat de Girona – Escola Politècnica Superior

COMPUTADORS Curs 2003-04 14 de Gener.

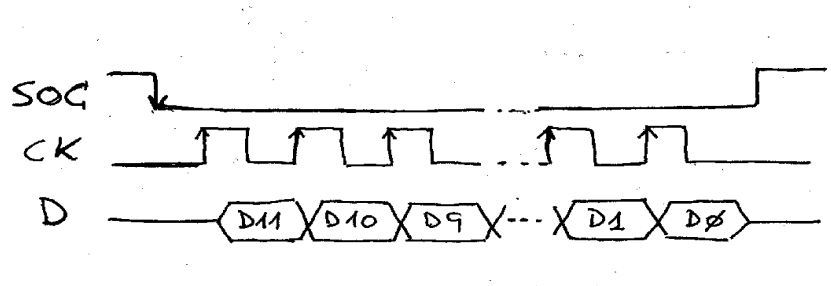
Examen Problemes. **Feu els exercicis en fulls separats.**

Les notes es publicaran el dimecres 21 de gener, als taulells del edifici PII i a http://eia.udg.es/computadors. La revisió de l'examen es realitzarà el divendres 23 de gener a les 10 hores (ja notificarem l'aula per a la revisió).

1) Bingo & Speech.

Després d'aquestes festes, i en plena època de rebaixes, ens han encarregat el disseny d'un sistema de transmissió d'àudio digital. L'esmentat sistema estarà format per un microcontrolador i realitzarà les tasques de gestió d'un convertidor analògic digital, compressió del senyal i transferència d'aquest.

Per a la conversió farem servir un convertidor analògic-digital d'aproximacions successives de dotze bits amb interfície sèrie. Aquest dispositiu (que en tenim 'en stock') té una entrada analògica, a la que s'hi pot connectar directament un micròfon, i tres línies digitals (compatibles 5V TTL) anomenades 'SOC' (Start Of Conversion), 'CK' (rellotge) i 'D' (dada). El funcionament d'aquest és molt simple: Un flanc descendent en el pin 'SOC' inicia un cicle de conversió, i els dotze bits són desplaçats un darrera l'altre, pel pin 'D', a cada flanc ascendent del senyal 'CK' (és aconsellable llegir-los en el flanc descendent). El primer bit desplaçat és el més significatiu. El cicle de conversió s'acaba amb un nivell alt del senyal 'SOC' (després de dotze cicles de rellotge) tal i com s'ilustra a la següent figura:



Com que el senyal a comprimir prové de la 'parla humana' i no cal una gran qualitat fixarem la banda passant a 4KHz i pel teorema del mostreig caldrà efectuar la conversió del senyal a 8Khz.

Per a la compressió d'aquest senyal farem servir la coneguda compressió logarítmica. Aquesta es basa en el fet que la percepció auditiva humana respon a un patró logarítmic. Per a un 'increment de sensació' en amplituts baixes és necessari un menor increment de pressió sonora que en amplituts altes; o, dit d'una altra manera, per percebre un increment de so a un nivell alt s'ha d'augmentar considerablement aquest nivell. Aquest fenòmen fóu aplicat des dels inicis de la 'timofonía' amb èxit i s'han establert certs 'standards' coneguts sota els noms de 'llei-A' i 'llei-μ'. Aquesta conversió l'efectuarem mitjançant dues taules de logaritmes (per la llei 'A' i 'μ') pre-calculades que hi farà correspondre a cada valor de dotze bits un resultat de vuit bits.

La selecció de la compressió s'efectuarà a través d'un interruptor extern.

Com que estem en una primera fase de desenvolupament, la transmissió d'aquest senyal es farà 'en paralel' a través d'un port de vuit bits (per exemple P1) sense cap tipus de control ni protocol (simplement sempre hi haurà present l'últim valor comprimit).

Malauradament al mirar al calaix dels components només hi hem trobat dispositius del tipus 8031, EPROMS de 8 Kbytes, RAMs de 2Kbytes i molts circuits TTL de diversos tipus.

A) Fes un diagrama del sistema físic que permet resoldre el problema. Comenteu de forma breu els diferents dispositius que hi intervenen i les connexions entre ells. Cal optimitzar en cost.

Informació: el programa principal serà el següent.

```
ORG 00h
     AJMP inici

;Aqui van els vectors d'interrupció

taula_llei_a:  DB 0,1,2,3,4.....
                  ....
               DB 254,255,255,255
taula_llei_mu: DB 0,1,2,3,4....
                  ....
               DB 254,254,255,255

;Aqui van totes les subrutines (incloses les de servei a la interrupció)

inici:    CALL ini_timer
          CALL ini_int
          JMP  $
END
```

Aquest codi efectua les inicialitzacions necessàries i resta executant un bucle infinit. Hi haurà una interrupció periòdica que s'encarregarà de llegir una paraula del convertidor, comprimir el valor i treure el resultat de la següent manera:

```
RSI_periodica: %llegeix_conversor
               %comprimeix
               %treu_valor
RETI
```

B) Implementeu en llenguatge assemblador les rutines 'ini_timer' i 'ini_int' de manera que es produeixin vuit mil interrupcions cada segon (suposeu que tenim un cristall de 11,0592 Mhz) i siguin ateses per 'RSI_periodica'. Comenteu tot el necessari (valors, registres, vectors) perque funcioni.

C) Implementeu en llenguatge assemblador la porció de codi que gestiona el conversor ('%llegeix_conversor', indicada com una macro). Indiqueu quin 'lligam' hi ha amb el codi que vé després (no cal que penseu amb 'retorn de valors').

D) Implementeu en llenguatge assemblador la porció de codi efectua la compressió ('% comprimeix', indicada com una macro). Indiqueu quin 'lligam' hi ha amb el codi precedent i el posterior (no cal que penseu amb 'pas de paràmetres' ni 'retorn de valors').

2)El 'nyafro'

Volem construir una sèrie de 'filtres' per aplicar a certes informacions que tenim emmagatzematdes (correu, fitxers...). Aquests petits programes ens permetran 'embrutir' d'una manera controlada la informació (sense que es pugui arribar a parlar de criptografia) de forma que el resultat sigui únicament 'soroll' als ulls del que no conegui la clau amb que ha estat 'desordenat'. El procés ha de ser 'reversible' i poder restituir el missatge original amb el tractament adeqüat. La estratègia a seguir és aplicar la funció 'OR-Exclusiva' sobre les totes sub-cadenes, de mida igual a la d'una paraula clau (bit a bit), d'una cadena. Donada una paraula clau, que limitarem a quatre caracters, i una cadena de mida 'n', efectuarem la XOR dels primers quatre caracters de la cadena amb els de la paraula clau. Seguidament efectuarem la mateixa operació amb els caracters 'del segon al cinquè' i així successivament fins arribar al final de la cadena.

A) Implementeu la acció 'nyafra' a la que s'hi passa una cadena (acabada amb el caracter NULL) i una clau, ambdues passades per referència. Compteu amb la següent definició:

```
.data
cadena DB 'Hola que tal',0
clau DB 8,0,8,6
```

Aquesta funció serà cridada de la forma:

```
PUSH  clau     ;crido a nyafra(cadena, clau). Paràmtres per referència
PUSH  cadena
CALL      nyafra
```

Recomanació: Doneu per fet que existeix (i crideu-la) una funció anomenada 'xor_a' a la que s'hi passen les adreces de dues cadenes de quatre bytes i efectua la XOR bit a bit entre els bytes de les cadenes.

B) Implementeu la funció xor_a' descrita anteriorment. Sigueu conseqüents ambla crida efectuada i tingueu present que s'ha de modificar la sub-cadena i no la clau.

4) L'Assemblador.

Codifiqueu les instruccions 'MOV' del següent codi i8086 indicant, en hexadecimal, quina és la imatge en memòria (deixeu tres bytes buits pel 'LEA').

```
MOV  AX,0
MOV  ES,AX
MOV  DI,128*4
LEA  AX,la_meva_rutina
MOV  ES:[DI],AX
MOV  ES:2[DI],CS
```

REGISTER	W=1	CODE W=0
AL	AX	000
BL	BX	011
CL	CX	001
DL	DX	010
AH	SP	100
BH	DI	111
CH	BP	101
DH	SI	110

SEGREG	CODE
CS	01
DS	11
ES	00
SS	10

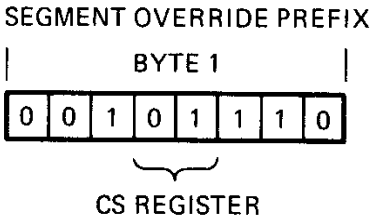


FIGURE 3-7 Instruction codes for 8086 registers.

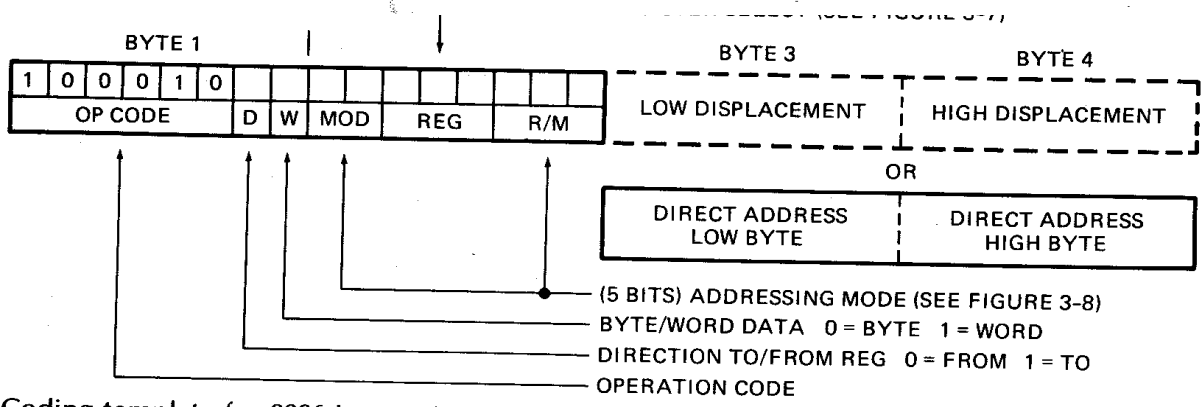


FIGURE 3-6 Coding template for 8086 instructions which MOV data between registers or between a register and a memory location.

MOD / R/M	00	01	10	11	
				W = 0	W = 1
000	[BX] + [SI]	[BX] + [SI] + d8	[BX] + [SI] + d16	AL	AX
001	[BX] + [DI]	[BX] + [DI] + d8	[BX] + [DI] + d16	CL	CX
010	[BP] + [SI]	[BP] + [SI] + d8	[BP] + [SI] + d16	DL	DX
011	[BP] + [DI]	[BP] + [DI] + d8	[BP] + [DI] + d16	BL	BX
100	[SI]	[SI] + d8	[SI] + d16	AH	SP
101	[DI]	[DI] + d8	[DI] + d16	CH	BP
110	d16 (direct address)	[BP] + d8	[BP] + d16	DH	SI
111	[BX]	[BX] + d8	[BX] + d16	BH	DI

MEMORY MODE — REGISTER MODE

d8 = 8-bit displacement d16 = 16-bit displacement

FIGURE 3-8 MOD and R/M bit patterns for 8086 instructions. The effective address (EA) produced by these addressing modes will be added to the data segment base to form the physical address, except for those cases where BP is used as part of the EA. In that case the EA will be added to the stack segment base to form the physical address. You can use a segment-override prefix to indicate that you want the EA to be added to some other segment base.