UNIVERSITI SAINS MALAYSIA

Peperiksaan Semester Pertama
Sidang 1990/91

Oktober - November 1990

EET 410 - Pemprosesan Isyarat Digit

Masa : [3 jam]

ARAHAN KEPADA CALON:

Sila pastikan bahawa kertas peperiksaan ini mengandungi 7 muka surat beserta LAMPIRAN (1 muka surat) bercetak dan LIMA (5) soalan sebelum anda memulakan peperiksaan ini.

Jawab EMPAT soalan.

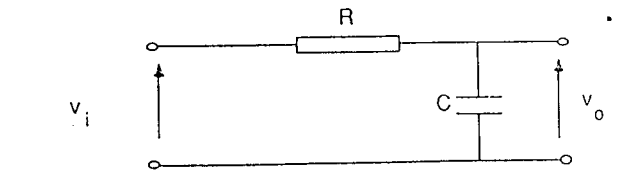
Agihan markah bagi setiap soalan diberikan di sut sebelah kanan sebagai peratusan daripada markah keseluruhan yang diperuntukkan bagi soalan berkenaan.

Jawab kesemua soalan di dalam Bahasa Malaysia.

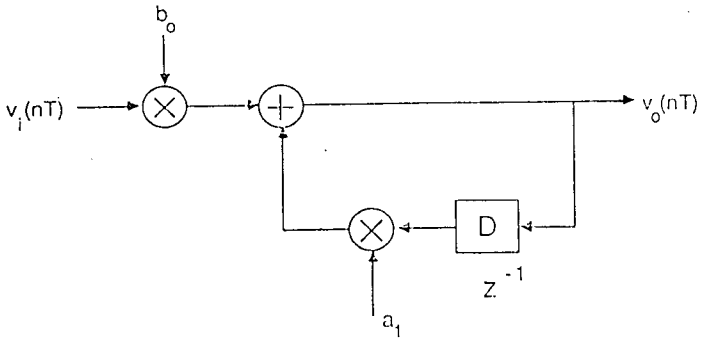
...2/-

1. (a) Buktikan bahawa bentuk setara masa-diskrit bagi penuras analog RC (Rajah 1(a)) adalah seperti yang diberikan dalam Rajah 1(b) Nyatakan semua andaian.

(30%)



(a)



(b)

Rajah 1

(b) Dalam sesuatu sistem pemprosesan isyarat digit, biasanya penuras analog masih diperlukan. Terangkan jenis dan fungsi-fungsi penuras analog tersebut.

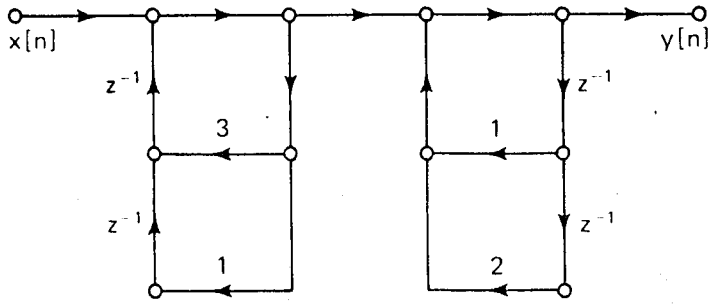
(10%)

(c) Suatu jenis penuras digit boleh dilaksanakan seperti dalam Rajah 2.

(i) Tuliskan persamaan beza bagi y(n).
(ii) Dapatkan jelmaan Z bagi y(n).
(iii) Seterusnya, dapatkan fungsi pindah bagi penuras tersebut.
(iv) Perlaksanaan dalam Rajah 2 memerlukan 4 daftar storan. Dapatkah jumlah ini dikurangkan ?

(40%)

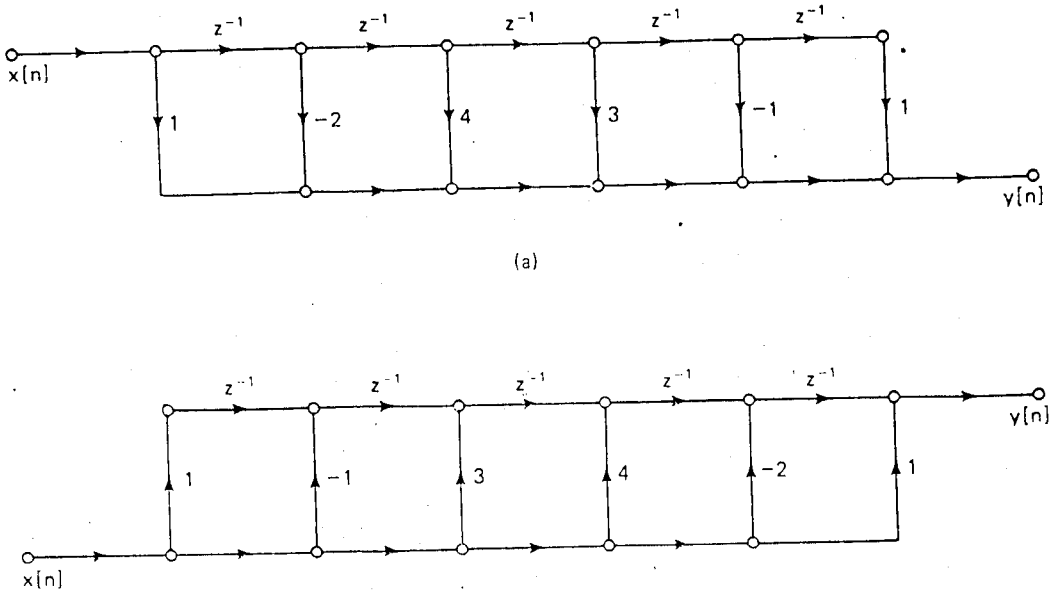
...3/-



Rajah 2

(d) Berikan sambutan dedenyut bagi penuras-penuras dalam Rajah 3 (a) dan 3(b).

(20%)



Rajah 3

2. (a) Dengan bantuan gambarajah, huraikan tatacara merekabentuk penuras FIR menerusi teknik tingkap dan teknik persampelan frekuensi.

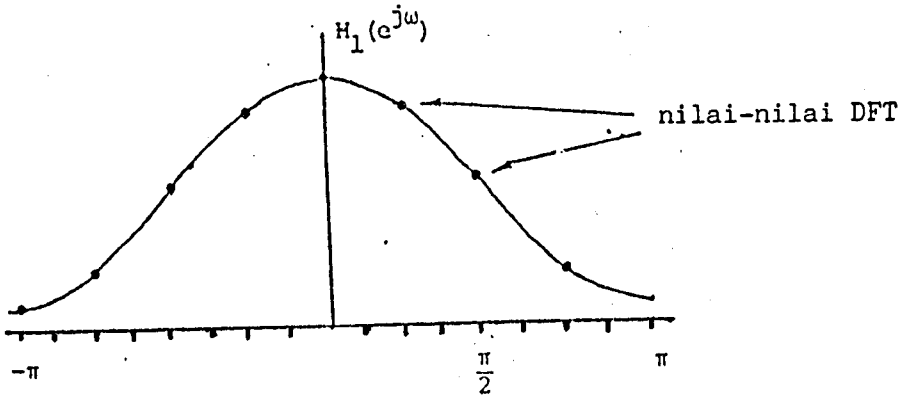
(30%)

...4/-

(b) $h_1(n)$ dan $h_2(n)$ adalah sambutan-sambutan dedenyut bagi dua penuras FIR 16-titik. Hubungan antara jujukan-jujukan tersebut adalah dalam bentuk anjakan pekeliling 8-titik, iaitu:

$$h_2(n) = <h_1(n-8)> \bmod 16$$

Jelmaan Fourier bagi $h_1(n)$ adalah seperti dalam Rajah 4.



Rajah 4 - Jelmaan Fourier untuk $h_1(n)$

(i) Berikan hubungan antara DFT{$h_1(n)$} dengan DFT{$h_2(n)$} , dan tunjukkan bahawa magnitud bagi kedua-duanya adalah sama.

(20%)

(ii) Adakah sambutan frekuensi bagi $H_2(e^{j\omega})$ juga sama dengan $H_1(e^{j\omega})$.

(20%)

(c). Sambutan dedenyut bagi sebuah sistem digit lelurus masa tak varian adalah h(n) = {4, 3, 1, 0}. Jujukan x(n) = {5,2,3,0} dimasukkan ke masukan sistem tersebut. Dapatkan hasil pelingkaran antara h(n) dengan x(n) menerusi cara berikut:

(i). $y(n) = h(n) * x(n)$
(ii). $Y(z) = H(z) \cdot X(z)$

(30%)

3. (a) Terbitkan persamaan kupu-kupu ('butterfly') asas bagi algoritma FFT 'Decimation-in-Time'.

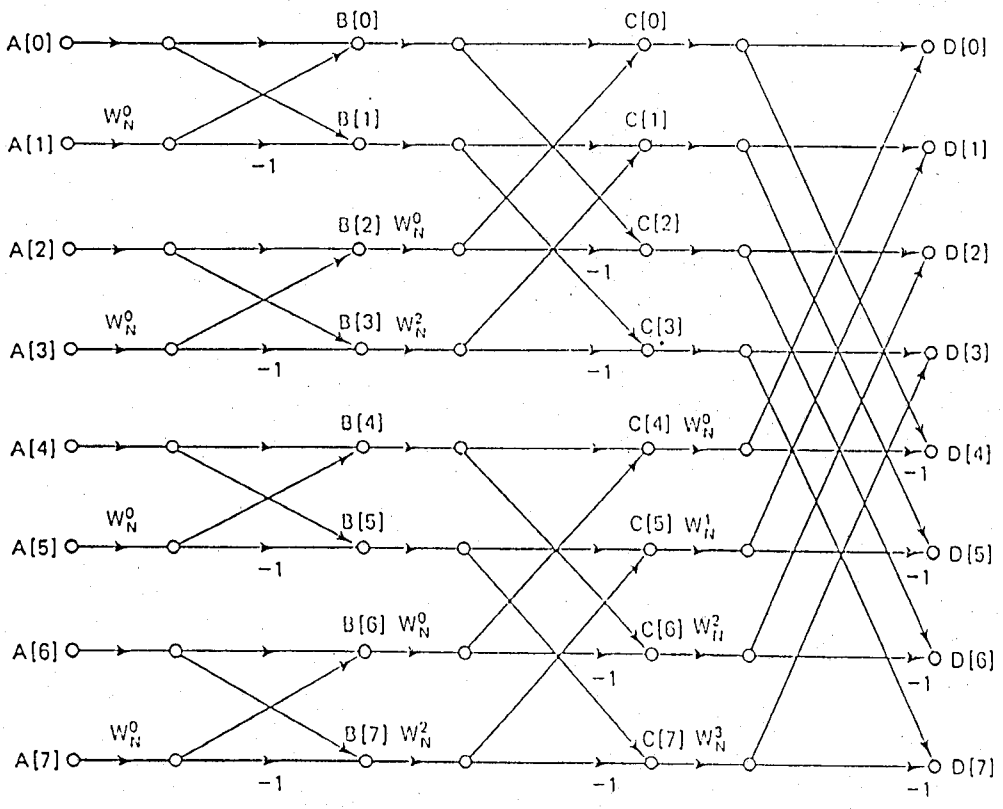
(30%)

(b) Aturcara DFT boleh juga digunakan untuk mengira nilai-nilai DFT songsang. Cara yang mudah ialah dengan menyusun jujukan sampel frekuensi masukan tersebut dalam susunan tertentu. Tunjukkan susunan tersebut.

(10%)

(c) Geraf alir isyarat dalam Rajah 5 digunakan untuk menjelmakan jujukan frekuensi X(k) balik semula ke domain masa (iaitu x(n)). Bagaimanakah elemen-elemen jujukan X(k) disusun di dalam tatasusunan A[r]. Tunjukkan juga bagaimana jujukan keluaran x(n) diperolehi dari D[r].

(20%)



Rajah 5

(d) Tanpa mengira nilai-nilai dalam tatasusunan B[.] dan C[.], lakarkan jujukan D[r], r = 0,1,..,7; jika jujukan masukan $x(n) = (-W_N)^n$, n = 0,1,..,7.

(20%)

(e) Dalam Rajah 5, sekiranya keluaran Jelmaan Fourier yang diperolehi ialah: X(k) = 1, k = 0,1,..,7 ; lakarkan jujukan C[r], r = 0,1,..,7.

(20%)

4. (a) Fungsi utama penjelmaan DFT ialah untuk mengesan dan mengukur komponen-komponen frekuensi bagi sesuatu isyarat. Sebagai contoh, katakan suatu penyahkod DTMF ("Dual Tone Multi-Frequency") bagi telefon jenis "push-button" hendak dilaksanakan dengan menggunakan kaedah DFT. Jadual 1 menunjukkan frekuensi-frekuensi yang digunakan oleh butang-butang telefon berkenaan. Umpamanya, bagi butang 1, isyarat 697Hz dan 1209Hz dikeluarkan; manakala 697Hz dan 1336Hz digunakan bagi butang 2, dan sebagainya.

Butang	Ton Rendah	Ton Tinggi
1	697	1209
2	697	1336
3	697	1477
4	770	1209
5	770	1336
6	770	1477
7	852	1209
8	852	1336
9	852	1477
*	941	1209
O	941	1336
#	941	1477

Jadual 1 - Frekuensi-frekuensi yang digunakan oleh butang-butang telefon

(i) Tentukan frekuensi persampelan minimum yang mencukupi (nilai kHz terdekat).

(ii) Tentukan resolusi minimum yang diperlukan.

(iii) Dari (i) dan (ii), dapatkan jumlah titik jelmaan minimum yang diperlukan.

(iv) Panjang minimum tersebut mungkin tidak memadai. Mengapa?

(v) Adakah penggunaan DFT dalam kes ini praktik atau tidak? Beri sebab-sebab.

(50%)

(b) (i) FFT boleh digunakan sebagai alat untuk melaksanakan proses pelingkaran. Dalam keadaan apakah kaedah ini boleh menguntungkan?

(5%)

(ii) Dengan bantuan gambarajah, huraikan kaedah tindih-simpan ("overlap-save") .

(15%)

(iii) Kaedah tindih-simpan digunakan untuk melingkar satu jujukan tak-terhingga x(n) dengan penuras (FIR) P-titik h(n). Panjang FFT yang digunakan ialah L-titik ($L = 2^V$). Dapatkan suatu ungkapan bagi jumlah operasi darab kompleks yang diperlukan bagi setiap sampel yang dihasilkan.

(30%)

5. (a) Tuliskan aturcara TMS32010 untuk melaksanakan penuras-penuras berikut:

(i) IIR Bentuk-Terus II, tertib kedua.

(ii) FIR N-titik (Aturcara Bergelung).

(40%)

(b) Laksanakan algoritma FFT 8-titik "decimation-in-time" dengan menggunakan pemproses TMS32010.

Butir-butir yang dikehendaki ialah:

(i) Carta alir bagi algoritma.

(ii) Pembahagian ingatan.

(iii) Cara perlaksanaan operasi pembalikan bit ("bit-reversal").

(iv) Penjanaan alamat kupu-kupu.

(v) Contoh aturcara.

(60%)

- oooOooo -

AUXILIARY REGISTER AND DATA PAGE POINTER INSTRUCTIONS

MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER (15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0)
LAR	Load auxiliary register	1	1	0 0 1 1 0 0 0 R ← D →
LARK	Load auxiliary register immediate	1	1	0 1 1 1 0 0 0 R ← K →
LARP	Load auxiliary register pointer immediate	1	1	0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 K
LDP	Load data memory page pointer	1	1	0 1 1 0 1 1 1 1 1 ← D →
LDPK	Load data memory page pointer immediate	1	1	0 1 1 0 1 1 1 0 0 0 0 0 0 0 0 K
MAR	Modify auxiliary register and pointer	1	1	0 1 1 0 1 0 0 0 1 ← D →
SAR	Store auxiliary register	1	1	0 0 1 1 0 0 0 R 1 ← D →

BRANCH INSTRUCTIONS

MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER (15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0)
B	Branch unconditionally	2	2	1 1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BANZ	Branch on auxiliary register not zero	2	2	1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BGEZ	Branch if accumulator ≥ 0	2	2	1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BGZ	Branch if accumulator > 0	2	2	1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BIOZ	Branch on $\overline{BIO}$ = 0	2	2	1 1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BLEZ	Branch if accumulator ≤ 0	2	2	1 1 1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BLZ	Branch if accumulator < 0	2	2	1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BNZ	Branch if accumulator ≠ 0	2	2	1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BV	Branch on overflow	2	2	1 1 1 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BZ	Branch if accumulator = 0	2	2	1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
CALA	Call subroutine from accumulator	2	1	0 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0
CALL	Call subroutine immediately	2	2	1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
RET	Return from subroutine	2	1	0 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 1

T REGISTER, P REGISTER, AND MULTIPLY INSTRUCTIONS

MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER (15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0)
APAC	Add P register to accumulator	1	1	0 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1
LT	Load T register	1	1	0 1 1 0 1 0 1 0 1 ← D →
LTA	LTA combines LT and APAC into one instruction	1	1	0 1 1 0 1 1 0 0 1 ← D →
LTD	LTD combines LT, APAC, and DMOV into one instruction	1	1	0 1 1 0 1 0 1 1 1 ← D →
MPY	Multiply with T register; store product in P register	1	1	0 1 1 0 1 1 0 1 1 ← D →
MPYK	Multiply T register with immediate operand; store product in P register	1	1	1 0 0 ← K →
PAC	Load accumulator from P register	1	1	0 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 0
SPAC	Subtract P register from accumulator	1	1	0 1 1 1 1 1 1 1 1 1 0 0 1 0 0 0 0 0

CONTROL INSTRUCTIONS

MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER (15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0)
DINT	Disable interrupt	1	1	0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1
EINT	Enable interrupt	1	1	0 1 1 1 1 1 1 1 1 0 0 0 0 0 1 0
LST	Load status register	1	1	0 1 1 1 1 0 1 1 1 ← D →
NOP	No operation	1	1	0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0
POP	Pop stack to accumulator	2	1	0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1
PUSH	Push stack from accumulator	2	1	0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0
ROVM	Reset overflow mode	1	1	0 1 1 1 1 1 1 1 1 1 0 0 0 1 0 1 0
SOVM	Set overflow mode	1	1	0 1 1 1 1 1 1 1 1 1 0 0 0 1 0 1 1
SST	Store status register	1	1	0 1 1 1 1 1 1 0 0 1 ← D →

I/O AND DATA MEMORY OPERATIONS

MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER (15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0)
DMOV	Copy contents of data memory location into next location	1	1	0 1 1 0 1 0 0 1 1 ← D →
IN	Input data from port	2	1	0 1 0 0 0 ← PA → 1 ← D →
OUT	Output data to port	2	1	0 1 0 0 1 ← PA → 1 ← D →
TBLR	Table read from program memory to data RAM	3	1	0 1 1 0 0 1 1 1 1 ← D →
TBLW	Table write from data RAM to program memory	3	1	0 1 1 1 1 1 0 1 1 ← D →

ACCUMULATOR INSTRUCTIONS

MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER (15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0)
ABS	Absolute value of accumulator	1	1	0 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0
ADD	Add to accumulator with shift	1	1	0 0 0 0 ← S → 1 ← D →
ADDH	Add to high-order accumulator bits	1	1	0 1 1 0 0 0 0 0 1 ← D →
ADDS	Add to accumulator with no sign extension	1	1	0 1 1 0 0 0 0 1 1 ← D →
AND	AND with accumulator	1	1	0 1 1 1 1 0 0 1 1 ← D →
LAC	Load accumulator with shift	1	1	0 0 1 0 ← S → 1 ← D →
LACK	Load accumulator immediate	1	1	0 1 1 1 1 1 1 1 0 ← K →
OR	OR with accumulator	1	1	0 1 1 1 1 0 1 0 1 ← D →
SACH	Store high-order accumulator bits with shift	1	1	0 1 0 1 1 ← X → 1 ← D →
SACL	Store low-order accumulator bits	1	1	0 1 0 1 0 0 0 0 1 ← D →
SUB	Subtract from accumulator with shift	1	1	0 0 0 1 ← S → 1 ← D →
SUBC	Conditional subtract (for divide)	1	1	0 1 1 0 0 1 0 0 1 ← D →
SUBH	Subtract from high-order accumulator bits	1	1	0 1 1 0 0 0 1 0 1 ← D →
SUBS	Subtract from accumulator with no sign extension	1	1	0 1 1 0 0 0 1 1 1 ← D →
XOR	Exclusive OR with accumulator	1	1	0 1 1 1 1 0 0 0 1 ← D →
ZAC	Zero accumulator	1	1	0 1 1 1 1 1 1 1 1 0 0 0 1 0 0 1
ZALH	Zero accumulator and load high-order bits	1	1	0 1 1 0 0 1 0 1 1 ← D →
ZALS	Zero accumulator and load low-order bits with no sign extension	1	1	0 1 1 0 0 1 1 0 1 ← D →