

마이크로프로세서를 이용한 디지털 필터 設計研究 (Research on Microprocessor Based Digital Filter Design)

李 幸 世*
(Lee, Haing Sei)

要 約

디지털 필터를 마이크로프로세서를 이용하여 設計하였다. 포커스 數를 使用하여 數를 表示하므로써 乘除算을 高速化하였고, 加減算은 換算表를 活用하여 處理하였다. 이리하여 三次의 低域通過 디지털 필터의 샘플이 500 (samples/sec)까지 處理 되었다.

Abstract

Microprocessor based digital filter was designed using focus number system in order to increase the multiplication speed of microprocessor program. Addition and subtraction program was treated using look up table. The sampling speed was improved up to 500 samples per seconds on the third-order low-pass digital filters.

1. 序 論

마이크로프로세서의 出現으로 回路나 시스템의 設計分野에 大端한 變化가 나타나게 되었다. 즉, 從來 하드웨어의 改善 및 變化에 依해서만 性能을 發揮하던데서 부터 소프트웨어나 펌웨어 (firmware)를 同時에 考慮하여 하드웨어의 改造 없이도 傳達函數나 特性을 소프트웨어의 變化 乃至는 制御信號에 依해서 變化 可能하다는 것이다. 예를 들어 發振回路로 設計하였는데 그 部分에 低域通過 필터를 代置코져 할 경우 發振回路를 떼어내고 필터를 設計하여 代置하였던 從來의 方式에서 부터 마이크로프로세서를 使用할 경우 프로그램만을 바꾸어 주거나, 이미 저장되어 있는 프로그램을 불러내는 命令으로 足하다는 點이다. 마이크로프로세서를 使用할 境遇에는 거의 願하는 傳達函數를 全部 實現할 수 있으며, 價格面에서도 壹弗에서 부터 數拾弗 程度이므로 큰 問題가 안된다.

그러나 現在 가장 廣範하게 使用되는 8 비트형 마이크로프로세서는 大部分 n-MOS型으로 되어 있고 速度面에서 가장 큰 問題를 가지고 있다.

한편, 디지털 필터는 信號處理 過程에서 必要不可缺한 것으로 일찍부터 研究되었으나 아직까지 많은 問題들이 남아 있다. TTL IC를 組立하여 實現할 境遇는 速度는 빠르지만, 傳達函數를 變化시키고저 할때는 大部分 새로운 回路로 代替하여야 될 境遇가 發生하여 回路의 最少化나 最適化가 無意味하여 지게 된다. 그래서 낮은 周波數나 低速의 信號에는 適應性이 좋은 마이크로프로세서가 要求 되어진다. 몇몇의 마이크로프로세서를 利用한 디지털 필터의 設計가 試圖되었으나 速度面에서의 精度面에서 커다란 問題를 內包하고 있다. [1, 2] 問題는 마이크로 프로세서에서 加減算은 두 세계의 命令語로 遂行되지만, 乘除算에서는 乘除數의 비트數 만큼의 시프트 (shift)와 加減算 등이 遂行되어야 解結된다. 특히 精度를 높이기 爲해서는 四則 全部가 複雜하여 지고 速度의 低下가 甚하여 진다.

또한 8 비트형 마이크로프로세서의 경우 演算이 8 비트씩 可能하기 때문에 數는 結局 8 비트의 倍數가 되는 것이 普通이다. 速度面에서는 可能的 限(비트數)을 줄이는 것이 좋고, 精度面에서는 비트數를 늘이는 것이 좋다. 速度를 높이기 위해서 비트數를 줄이면, 오버플로우 (over flow) 現象, 演算誤差, 係數 및 入出力의 誤差가 顯著하여 지고, 아나로그 (analog) 信號를 入力으로 使用할 境遇에는 오프셋(off-

* 正會員, 亞洲工科大学 電子工學科
(Dept. of Electronics Engineering Ajou Institute of Technology)
接受日字: 1979年 5月 4日

set) 誤差가 重要한 雜音源이 되기도 한다. 이를 充分히 考慮하면 速度가 낮아서 쓸모없게 되어 버린다.

이와 같은 問題의 解決이 필타設計의 先決問題로 되어 있다.

2. 포커스 (Focus) 數

一般的으로 컴퓨터에서는 實數를 固定 小數點數와 浮動 小數點數로 表示하고 있다. 예를 들면 53.817 이라는 數가 있으면, 다음과 같이 $53.817 = 0.53817 \times 10^2$ 로 表示하자. 왼쪽의 數는 固定 小數點數이고, 오른쪽은 浮動 小數點數이다. 왼쪽의 數는 小數點의 자리가 固定된 數이고, 오른쪽의 數는 指數의 값에 따라 浮動하게 된다. 固定 小數點數는 浮動 小數點數에 比해서 速度는 빠른 便이나 오버플로우나 언더플로우가 쉽게 나타나서 誤差源이 되며 演算 過程에서 더 많은 誤差를 나타낸다. 여기에서는 小數部를 없애고, 指導部만을 使用하는 數를 生覺하여 보자. 萬一에 X라는 값을 다음과 같이 表示하면

$$X = (-1)^s \cdot a^{(P+1)} \cdot N \dots\dots\dots (1)$$

단, s 나 p 는 0 이나 1 이고, N 는 自然數이고, $a > 1$ 이라 하자. 任意的 實數 X가 주어지면 여기에 對應되는 S, P, N이 求하여 진다. 즉,

$$N = (-1)^{(P+1)} \log_a \{ (-1)^s \cdot X \} \dots\dots\dots (2)$$

식(2)에서 $(-1)^s \cdot X$ 는 恒常 陽數이어야 하므로 X가 陽數이면 S=0, X가 陰數이면 S=1이 되며, 右邊 값이 또한 恒常 陽數이어야 하므로 對數의 값이 陽數이면, P=1, 對數의 값이 陰數이면, P=0이 되어야 한다. 결국 $(-1)^s$ 는 X의 符號와 一致하고 $(-1)^{(P+1)}$ N은 X의 絶對值의 a에 對한 對數值이다.

즉 $(-1)^{(P+1)} \cdot N$ 는 하나의 整數 (Integer number)로 取扱하는 것이 便利하다. 特히 加減算을 爲해서는 陰數를 陽數의 2의 補數로 擇하는 것이 便利하다. 만일 a를 除外하고 表示하여, m비트의 二進數로 表示하면 다음과 같다.

$$F(X) = SP N_{m-3} N_{m-4} \dots\dots N_1 N_0 \dots\dots\dots (3)$$

이때 S는 X의 符號이고, P는 指數部의 符號인데 P=1은 陽數의 $N_{m-3} \dots\dots N_0$ 를 가지고, P=0는 2의 補數인 陰數의 $N_{m-3} \dots\dots N_0$ 를 가진다. a는 식(3)에 表示되지 않으므로 混同되어서는 안되며 반드시 使用者는 a 값을 밝혀야 한다. 또한 한 프로그램 안에서 任意로 바꾼다면 混亂을 가져오게 되어 피하는 것이 좋겠다. 몇개의 밑 a에 對한 換算表를 표 1에 提示하였다. 이와 같이 表示되는 數를 포커스數라 한

표 1. 포커스數와 실수의 환산표

(a는 Base number)

포커스수 (16진수)	실 수 (10진수)		
	$a = \sqrt{2}$	$a = \sqrt[12]{2}$	$a = \sqrt[8]{2}$
00	0.0625	0.0248	0.0039
01	0.0653	0.0263	0.0043
02	0.0682	0.0278	0.0046
03	0.0712	0.0295	0.0051
04	0.0743	0.0313	0.0055
05	0.0776	0.0331	0.0060
06	0.0811	0.0351	0.0066
07	0.0846	0.0372	0.0072
08	0.0884	0.0394	0.0078
09	0.0923	0.0417	0.0085
0A	0.1964	0.0442	0.0093
0B	0.1007	0.0468	0.0101
0C	0.1051	0.0496	0.0110
0D	0.1098	0.0526	0.0120
0E	0.1146	0.0557	0.0131
0F	0.1197	0.0590	0.0143
10	0.1250	0.0625	0.0156
11	0.1305	0.0662	0.0170

다. 參考文獻^[3]을 보면 잘 알 수 있겠지만 Lee와 Edgar의 경우는 本論文의 特殊한 境遇와 對應됨을 알 수 있다. 그들은 a를 2나 10으로 固定하고 N을 固定 小數點數로 看做하였다. 밑 $a = 2^{2^k} = \sqrt[2^k]{2}$ 인 경우에 表現되는 數와 2를 밑으로 하고 K번째 小數點을 擇한 경우 같은 數字로 表示된다. 따라서 演算의 境遇는 같은 方法이 使用될 수 있어, 性格이 다른 數의 表現이나 이름을 같이 포커스數로 하였다. 이와 같이 指數로 表現되는 포커스數는 乘除算이 加減算으로 解結되어 高速化 된다. 그러나 加減算의 境遇는 簡單하지 않다. 이를 解決하는 方法이 Lee와 Edgar에 의해서 換算法 (look up table)을 利用하는 方法이 提示되어 本論文에서는 이를 修正하여 使用하였다.

즉 $x > y$ 의 境遇,

$$a^x + a^y = a^z = a^{x+F(x-y)} \dots\dots\dots (4)$$

$$F(x-y) = \log_a(1+a^{-(x-y)}) / \log_a \dots\dots\dots (5)$$

$$a^x - a^y = a^w = a^{x+G(x-y)} \dots\dots\dots (6)$$

$$G(x-y) = \log_a(1-a^{-(x-y)}) / \log_a \dots\dots\dots (7)$$

이와 같은 函數 $F(N)$, $G(N)$ 을 計算하고 圖表化하여 이를 찾아서 加算하면 速度를 增加할 수 있고 또한 演算誤差도 줄일 수 있게 된다. 필터 設計時 다이나미한 데이터의 變化를 予測하여 이에 適合한 밑 a 를 選擇할 수 있도록 표 2 에 8 비트의 境遇 表示範圍 및 相對誤차를 表示하였다. 여기서 必要도 하는 範圍에 다른 任意的 適應이 밑 (Base number) 를 바꾸므로 可能함을 알 수 있다. 물론 이 數의 體系가 自動制御나 通信工學등에 活用될 수 있음을 말할 나위도 없다.

표 2. 밑 (Base number) 과 表示可能值과 相對誤差

a	a^{-64}	a^{63}	$\frac{a-1}{2a}$
$\sqrt[8]{2}$	0.0039	234.75	4.1
$\sqrt[9]{2}$	0.0072	128.00	3.7
$\sqrt[10]{2}$	0.0118	78.79	3.3
$\sqrt[11]{2}$	0.0177	52.97	3.1
$\sqrt[12]{2}$	0.0248	38.05	2.8
$\sqrt[13]{2}$	0.0330	28.76	2.6
$\sqrt[14]{2}$	0.0421	22.63	2.4
$\sqrt[15]{2}$	0.0520	18.38	2.3
$\sqrt[16]{2}$	0.0625	15.32	2.1
$\sqrt[17]{2}$	0.0736	13.05	2.0
$\sqrt[18]{2}$	0.0850	11.31	1.9
$\sqrt[19]{2}$	0.0968	9.96	1.8
$\sqrt[20]{2}$	0.1088	8.88	1.7

X의 絶對值가 a^N 일때, 라운드오프 (Round off) 誤차는 다음과 같이 表示된다.

$$E = \frac{a^N - a^{N-1}}{2} = a^N \left(\frac{a-1}{2a} \right) \dots\dots\dots(8)$$

따라서 라운드오프에 따른 相對誤차는

$$E_r = \frac{E}{a^N} = \frac{a-1}{2a} \dots\dots\dots(9)$$

로 되어 任意的 포커스數에서는 演算後에 發生하는 라운드오프 誤차가 數의 絶對值에는 無關하며 단지 밑 (Base number) 만의 函數임을 알 수 있다. 이에 反해서 絶對誤차는 定義하기가 困難하다. 만일 $a^N \cong 1$ 부근에서 絶對誤차는 相對誤차와 一致하며, 1 이하에서는 더욱 더 작아진다. 따라서 取扱할 數의 範圍가 決

定 되면 이에따라 演算誤差도 予測이 可能하여 진다.

그리고 固定 小數點數나 浮動 小數點에 比하여 乘除算의 速度는 比較할 수 없을 程度로 빨라진 反面, 加減算은 약간 늦어지고 數表가 必要하게 되었다. 8 비트의 境遇 最大로 256 개가 必要하므로 마이크로 프로세서의 記憶裝置에 저장할 때 價格上昇은 別로 큰 問題가 되지 않는다. 또 하나 長點은 a 가 클 경우 入力을 아나로그 量에서 A/D 變換을 行할 境遇에 오프 세트에 依해서 생기는 스텝의 變化가 充分히 微少하다. 즉 表示可能 最少值과 最大值 사이의 比가 固定 小數點數보다 大端히 棼을 알 수 있다. 그러나 a 가 작아지면 이는 成立하지 않는다.

3. 필터 實現을 爲한 소프트웨어

디지털 필터에는 FIR (Finite impulse response) 필터와 IIR (Infinite impulse response) 필터가 있고 이들은 加減算, 乘算, 遲延 (delay) 를 遂行하므로 實現할 수 있다. 그림 1 은 IIR 필터와 FIR 필터의 一般의인 例이다. 디지털 필터보다 아나로그 필터는 設計方法이 잘 整理되어 있어 普通 디지털 필터를 設計하고자 하면, IIR 필터의 境遇는 아나로그型 傳達函數에서부터 디지털型 傳達函數로 變換할 수 있다.

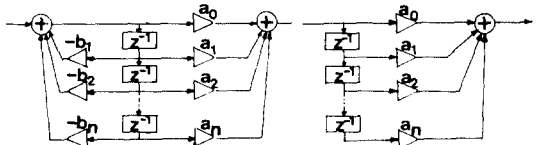


그림 1. IIR 필터와 FIR 필터의 例
Fig. 1. IIR filter and FIR filter.

(4) 低域通過 필터의 경우 通過帶域을 $f = 0$ 에서부터 $f = f_c$ 까지 通過시키고, $f = f_s$ 에서 그 이상, 즉 無限大까지 阻止시킨다고 하고, 이 周波數가 $\omega_c = 2\pi f_c = 1$ 로 正規化 되었다고 하자 디지털 필터의 경우 $\lambda = 0$ 에서부터 $\lambda = \lambda_c$ 까지 通過시키고, $\lambda = \lambda_s$ 에서 $\lambda = \pi$ 까지 阻止시킨다고 하면, 이들 사이에는 有限區間과 半無限區間과의 對應關係 (寫像, mapping) 가 되어 탄젠트 函數를 利用하여 關係 지을 수 있다.

$$\omega = \tan(\lambda/2) / \tan(\lambda_c/2) \dots\dots\dots(10)$$

또 $S = j\omega$ 와 $Z = e^{j\lambda}$ 로 놓으면,

$$S = \cot(\lambda_c/2) \frac{z-1}{z+1} \dots\dots\dots(11)$$

로 되어 식 (11)을 아나로그型 低域通過 필터의 傳達函數에 代入하면 디지털型 低域通過 傳達函數를 求할 수 있으며, 高域通過나 帶域通過등은 各各 對應關係를 考慮

하여 適合한 關係式을 求할 수 있다. IIR필타나 FIR필타가 大端히 큰 次數의 傳達函數를 가지게 되어 마이크로프로세서로 實現할 경우 IIR필타에 依存하게 되나 앞으로는 두가지 다 實現하게 될 것이다. 그림 1과 같이 表示되는 IIR필타는 다음과 같이 쓸 수 있다.

$$H(z) = \frac{\sum_{k=0}^M a_k z^{-k}}{1 + \sum_{k=1}^L b_k z^{-k}} \dots\dots\dots (12)$$

또 x_n 을 入力信號, y_n 을 出力信號라 하면,

$$y_n = \sum_{k=0}^M a_k X_{n-k} - \sum_{k=1}^L b_k y_{n-k} \dots\dots\dots (13)$$

로 計算된다. 식 (13)을 計算하는 代身에 그림 1과 같은 方法의 計算을 하는 것이 速度와 誤差를 줄일 수 있게 된다. z^{-1} 은 遲延回路로서 프로그램어블 타이머 (programmable timer)로서 實現된다. 最短時間은 全體 計算時間이내로 할 수 없으며 豫想되는 最大 時間보다 길어야 한다. 래치 (Latch) 回路는 記憶素子로 解決한다. 遲延時間이 지난 후 래치되는 값은 하나씩 場所를 모두 移動하여야 한다. 이와 같은 變數의 循環移動은 바람직하지 못하다. 그래서 遲延되어 래치되는 값은 連續하여 저장하고, 係數와 乘算에 便利하도록 세 번 反復 되도록 하였다. 그리고 첫 래치되는 값을 인덱스로 記憶시켰다가 係數와 乘算할 때 使用하였다. 이렇게 하여 얻어진 서브루틴 (subroutine) 프로그램과 시간측정 프로그램을 부록에 提示하였다.

4. 하이웨이 構成 및 實驗

本 實驗은 8비트型 마이크로프로세서로 가장 흔하게 使用되는 것중의 하나로서 美國 Intel社의 I8085 (CPU, 3MHz)를 使用하였다. I8355 (ROM, I/O)를 모니터用 記憶素子로 하였고, 두개의 I8155 (RAM, I/O, Timer)를 프로그램 開發用 記憶素子로 使用하였다. I8279를 키入力과 LED出力의 인터페이스 (Interface) 素子로 利用하였다. 이들의 블럭圖를 그리면 그림 2와 같다.

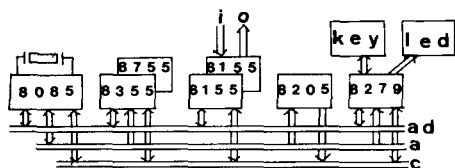


그림 2. 실험회로 블럭도
Fig. 2. Block diagram of experimental circuits

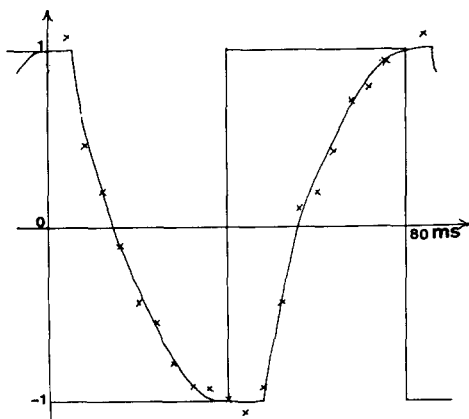


그림 3. 出力波型
Fig. 3. Output waveform.

이 시스템은 Intel社의 SDK-85를 母體로 하였으며, RAM 記憶素子와 EPROM 記憶素子등을 追加하여 實驗하였다. (5, 6) 實驗對像필타는 通過周波數가 $f_c = 1$ 까지이며, 阻止周波數가 $f_s = 2$ 부터 π 까지인 三次 低域通過 디지털 필타로서 그 傳達函數는 다음 式으로 주었다.

$$H(z) = \frac{0.125(1 + 2z^{-1} + z^{-2})}{1 - 0.707z^{-1} + 0.648z^{-2} - 0.136z^{-3}} \dots\dots\dots (14)$$

係數는 8비트 포커스數로 表示되었고, 이때 $a = 2^{\frac{1}{8}}$ 이었다.

絶對值가 1인 구형파 信號를 20샘플씩 各各 주었을 때 계산된 값이 그림 3에 表示되었다. 한개의 샘플을 處理하는 時間은 2msec를 超過하지 아니 하였다. 時間은 I8155의 프로그램어블 타이머를 使用하여 測定하였고, 오실로스코프로 確認하였다. 入力에 0은 存在하지 않으므로 0에 가까운 오프셋의 값에 따라 發生되는 出力誤差도 4%를 넘지 아니 하였다.

5. 結 論

마이크로프로세서의 시스템에 대한 適應性을 活用하고, 速度를 增加 시키기 위하여 포커스數를 導入하여 디지털 필타를 設計하고 實驗하여 三次 低域通過 필타에서 500 (샘플/초) 이상을 처리하였다. 앞으로 더욱 高速의 마이크로프로세서의 開發과 함께 다양한 應用分野가 予想되며, 특히 現時點에서는 低速適應性

制御分野에 活用이 期待 된다. 本 研究는 1978年度 文教部 研究費에 의해서 遂行되었으며 關係當局에 謝意를 表하는 바입니다.

參 考 文 獻

1. N. Ahmed and J. P. Jayapalan ; " On digital Filter Implementation via Microprocessors ", IEEE transaction on Industrial Electronics and Control Instrumentation, vol. IECI-23 , No. 3. Aug. 1976 .
2. T. Higuchi , T. Saito and A. Kanomata ; " mi - croprocessor - Based Digital Filter Program - med in a Block Diagram Language ", IEEE transaction on Industrial Electronics and Con - trol Instrumentation vol. IECI-24.No.3. Aug. 1977.
3. S. Lee. " Microcomputer Design and Applications ", Chap. 1. Academic Press, 1977.
4. A. Peled and B. Liu , "Digital Signal Proces - sing", John Wiley , 1976.
5. Intel , "SDK -85 User's Manual " Intel , 1977 .
6. Intel , "MCS -85 User's Manual " Intel , 1978 .

附 錄

(a) 常數 (加減算處理用)

記憶場所	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0800~0F	08	08	07	07	06	06	05	05	05	04	04	04	03	03	03	03
0810~1F	03	02	02	02	02	02	02	01	01	01	01	01	01	01	01	01
0820~2F	01	01	01	01	00	00	00	00	00	00	00	00	00	00	00	00
0830~3F	7F	1D	15	11	0E	0C	0A	09	08	07	06	06	05	05	04	04
0840~4F	03	03	03	02	02	02	02	02	02	01	01	01	01	01	01	01
0850~5F	01	01	01	01	01	00	00	00	00	00	00	00	00	00	00	00

註 0800~082 F까지는 住所値를 M 이라 하면 $N = M - 2800$, $F(N) = 8 \cdot \ln(1 + e^{-N/4}) / \ln 2$ 를 計算한 値이고 0830~085 F까지는 $F(N) = 8 \cdot \ln(1 - e^{-N/4}) / \ln 2$ 하였다.

(b) 加減算 副프로그램

記 憶 場 所	機 械 語	어 셈 블 러
0 8 6 0	26 08	FADD MVI H, 28
6 2	A8	XRA B
6 3	FA A8 08	JM LD
6 6	OE 2F	MVI C, 2F
6 8	A8	XRA B
6 9	FA 8C 08	JM L3
6 C	90	SUB B
6 D	FA 7B 08	JM LP
7 0	B9	CMP C
7 1	D2 88 08	JNC LA
7 4	6F	MOV L, A

7 5		ADD B
7 6		ADD M
7 7		RP
7 8		MVI A, 7 F
7 A		RET
7 B	LP	CMA
7 C		INR A
7 D		CMP C
7 E		JNC LB
8 1		MOV L, A
8 2		MOV A, B
8 3		AOD M
8 4		RP
8 5		MVI A, 7 F
8 7		RET
8 8		ADD B
8 9		RET
8 A		MOV A, B
8 B		RET
0 8 D 4		INR A
D 5		CMP C
D 6		JNC LB
D 9		ADD D
D A		MOV L, A
D B		MOV A, B
D C		SUB M
D D		RP
D E		MVI A, 0 0
E 0		RET
E 1	LS	CMP C
E 2		JNC LA
E 5		ADD D
E 6		MOV L, A
E 7		SUB D
E 8		ADD B
E 9		SUB M
E A		RM
E B		MVI A, 8 0
E D		RET

註; 加減코저 할 數가 Reg. A 나 B에 포커스數로 저장. 畵한 結果가 A에 저장됨.

(c) 連속된 變數를 加算하는 副프로그램

08F0	SUM	MOV A, M
F1	LPI	INR L
F2		MOV B, M
F3		MOV D, L
F4		PUSH D
F5		CALL FADD
F8		POP D
F9		DCR E
FA		JNZ LPI
FD		RET

註: Reg. E에 더할 數의 個數를 주며, Reg. H, L에 첫 숫자의 주소 저장요.
Reg. A에 저장되며, Reg. D에 주소, Reg. E에 루프값을 일시 저장함.

(d) 乘算副프로그램

2020	FMUL	XRA B
21		JM LM
24		XRA B
25		ADD B
26		SUI 40
28		RP
29		ADD A
2A		MVI A, 00
2C		RM
2D		MVI A, 7F
2F		RET
30	LM	XRA B
31		ADD B
32		SUI 40
34		RM
35		ADD A
36		MVI A, 80
38		RM
39		MVI A, FF
3B		RET

註: Reg. A와 B에 乘算値가 저장되며, 결과는 Reg. A에 저장됨.

(e) 連續된 變數列을 連續된 變數列과 乘해서 連續으로 저장하는 副프로그램.

203C	PRD	PUSH D
3D		MVI A, INDC

```

3 F      ADD E
4 0      MOV L, A
4 1      MOV B, M
4 2      MOV A, D
4 3      ADD E
4 4      MOV L, A
4 5      MOV A, M
4 6      CALL FMUL
4 9      POP D
4 A      MOV C, A
4 B      MVI A, INDR
4 D      ADD E
4 E      MOV L, A
4 F      MOV M, C
5 0      DCR E
5 1      JNZ PRD
5 4      RET
    
```

註; 被乘數의 INDEX가 있는 곳의 住所가 Reg. D에, 곱하는 個數가 Reg. E에 저장되어야함. 乘數의 저장장소는 INDEX+2000 부터이며 피승수는 L = INDC + 1 부터, 결과는 L = INDR + 1 부터 이다.

(f) 變數住所

2000	$X_1 X_2 X_3 X_4$	지연루프값	2010	계수 (분모) $D_1 = 0$	
01	$X_2 X_3 X_4 X_1$	"	11	"	D_2
02	$X_3 X_4 X_1 X_2$	"	12	"	D_3
03	$X_4 X_1 X_2 X_3$	"	13	"	D_4
04	$X_1 X_2 X_3 X_4$	"	14	계수 (분자) N_1	
05	$X_2 X_3 X_4 X_1$	"	15	"	N_2
06	$X_3 X_4 X_1 X_2$	"	16	"	N_3
07	$X_4 X_1 X_2 X_3$	"	17	"	N_4
08	$X_1 X_2 X_3 X_4$	"	18	승산결과저장장소 $X_1 \times D_1 = 0 = R_1$	
09	$X_2 X_3 X_4 X_1$	"	19	$X_2 \times D_2$	$= R_2$
0A	$X_3 X_4 X_1 X_2$	"	1A	$X_3 \times D_3$	$= R_3$
0B	$X_4 X_1 X_2 X_3$	"	1B	$X_4 \times D_4$	$= R_4$
0C	出力 = $S_1 + S_2 + S_3 + S_4$		1C	$X_1 \times N_1$	$= S_1$
0D	회환값 = $R_1 + R_2 + R_3 + R_4$		1D	$X_2 \times N_2$	$= S_2$
0E	入力		1E	$X_3 \times N_3$	$= S_3$
0F	변수 INDEX		1F	$X_4 \times N_4$	$= S_4$

(g) 샘플프로그램

2059		LXI SP, 28C0	A7	DCR B
5C		LXI H, 200E	A8	MVI A, 7F
5F		MVI A, FF	AA	OUT 28
61		OUT 28	AC	IN 2C
63		MVI A, 00	AE	LXI H, 2800
65		OUT 2C	B1	MOV M, A
67		MVI A, D8	B2	INR L
69		OUT 2D	B3	IN 2D
6B		MVI A, 40	B5	MOV M, A
6D		MOV M, A	B6	IN 28
6E		INR L	B8	INR L
6F		MVI B, 03	B9	MOV M, A
71	STRT	MOV M, B	BA	HLT
72		MOV L, B		
73		MOV M, A		
74		LXI D, 0004		
77		DAD D		
78		MOV M, A		
79		DAD D		
7A		MOV M, A		
7B		MVI L, 0F		
7D		MOV D, M		
7E	BGN	MVI E, 08		
80		CALL PRD		
83		MVI L, 1C		
85		MVI E, 04		
87		CALL SUM		
8A		LXI H, 200C		
8D		MOV M, A		
8E		MVI L, 18		
90		MVI E, 04		
92		CALL SUM		
95		LXI H, 200D		
98		MOV M, A		
99		INR L		
9A		MOV B, M		
9B		CALL FADD		
9E		LXI H, 2055		
A1		MOV M, A		
A2		OUT 29		
A4		MVI L, 0F		
A6		MOV B, M		

註. 이 프로그램은 지금까지의 모든 프로그램을 사용하여 計算時間을 測定하는 프로그램이다.