技術解說

16-Bit 마이크로프로세서 時代의 主役들 : 〔1〕 Intel 8086

金 悳 鎭

高麗大學校 工科大學 電子工學科 敎授 (工博)

마이크로프로세서는 이제 16-bit 時代로 접어 들었다. 이와 때를 같이 하여 마이크로프로세서의 應用範圍도 過去보다 훨씬 더 廣範圍하게 되어가고 있다. 本欄에서는 앞으로 16 - bit 마이크로프로세서들 中에서 Intel 8086, Zilog 8000 및 Motorola 68000 프로세서들의 特性을 차례로 連載하여 紹介하려고 한다.

1. Intel 8086

Intel 8080의 後繼者로서 開發된 것이 Intel 8086 인데 이것은 8080보다도 throughput의 處理能力이 約 10倍 程度 큰 것이다. 8086은 8080의 assembly語 software를 그대로 使用할 수 있도록 設計되어 있다. 이것은 8080 의 모든 register set와 instruction set가 8086에 包含되어 있기 때문에 可能하다. 本來 8086의 architecture 設計의 目標는 現存하는 8080의 性能을 擴大시킴과 아울러 다음의 諸特性을 附加시키는 것이었다. 즉 16-bit 演算, 符號붙은(signed) 8-bit 및 16-bit 演算 (乘算및 除算 包含), 能率的인 interrupt 可能한 byte-string 演算, bit 處理能力의 改善等이다.

이밖에도 64k bytes 以上을 直接 address 可能하고 multi-processing이 可能하게 되어 있다.

Intel 8086의 諸特性을 표 1에 列擧하였다.

Intel 8086은 HMOS (high performance MOS) 技術을 利用하여 製作한 processor 인데, 이 技術은 NMOS 方式에 있어 device 들의 칫수를 縮小시키므로써 channel 길이를 短縮하여 VLSI의 集積密度를 높이고, 動作速度를 빠르게 한 것이 特徵이라고 할 수 있다.

표 1에서 보는 바와 같이 8086은 約 1 mega byte 의 記憶容量을 가지며 64 k bytes 까지의 I/O port를 使用할 수 있다. Register 機構는 3個의 register file로 構成되어 있다. 數値演算과 論理演算에는 4個의 一般 register (general register) 가 使用되며, address 計算에는 2個의 16-bit pointer 와 2개의 16-bit index register 가 使用되고, addressing 範圍를 擴大하기 위하여 4개의 16-bit segment register 가 또 使用된다. Processor의 여러 가지 狀態를 나타내기 위하여 9개의 flag 이 使用된다.

8086의 充分한 instruction set 들은 廣範圍한 addressing mode 와 data transfer 演算을 可能하게 한다. 즉, 符號붙은 (signed) 경우와 符號붙지 않은 (unsigned) 경우의 8-bit 演算과 16-bit 演算, 論理演算, string 處理, control transfer 및 proccessor 制御 等을 可能케 한다. 그리고 外部와의 interface 에 있어서는 reset 順序, interrupt, multiprocessor 同期化, 및 resource-sharing 等도 包含될 수 있다.

표 1. Intel 8086의 特性

ALU width :		16 bits
Memory addressing capability :		1,048,576 bytes
Addressable I/O ports :		64 k
Process :		HMOS
Gate propagation delay :		≈ 2ns
Clock period :	200 ns standard	125ns selected
Memory access :	800 ns standard	500 ns selected
Relative performance :	8080 A, depending on character of program	7 ~ 12 times
Pins :		40
Power :		+5v

a. 記憶裝置의 構造

8086의 memory 構造는 2個의 部分으로 構成된다. 즉, 하나는 memory space 이고 다른 하나는 input/output space 이다. 모든 命令 code 와 operand 들은 memory space 안에 記憶되고, memory-mapped device 를 除外한 모든 周邊裝置 및 I/O 裝置들은 普通 I/O space 안에 記憶된다.

Memory space :

8086 memory는 8-bit byte 로 할때에 約 1 mega byte 이며, 16-bit word 로 形成하려면 인접한 2 byte씩을 묶어서 500 k words memory 로 使用할 수도 있다. 16-bit word 로 했을 때 홀수 byte address나 짝수 byte address로 指定할 수 있지만, 홀수 byte address로 指定할 때에는 2개의 hardware memory cycle을 必要로 하며, 짝수 byte address 로 指定할 때에는 1개의 memory cycle만 必要하다. 두 byte를 묶어서 쓸때에 16 bits 中 most significant 8 bits는 항상 높은 쪽 memory address에 기억된다.

16-bit 만으로는 64 k byte까지만 addressing 이 可能하므로 maga byte memory를 addressing 하려면 이를 위한 附加的인 addressing機構가 必要하다. 이를 위해서 8086 memory는 여러 개의 segment로 區分되어 있고 각 segment 는 最大 64 k byte 로 構成되는 것으로 생각한다. 각 segment 는 16으로 整除되는 address (즉 segment's address의 낮은 쪽 4-bit가 0이 되는 address) 에서 시작된다. 그리고 任意의 時間에 4개의 segment의 內容이 直接 addressing 될 수 있다. 이들 4개의 segment 는, 각각 current code segment extra segment 로 불리어지며 이들은 서로 重疊될 수도 있다. 각 segment 의 address 의 높은 쪽 16-bit는 지정된 16-bit register 인 segment register 에 기억되며 이를 segment address 라고 부른다. 特別한 경우로서, 이들 4개의 segment 가 모두 같은 address (즉 0番地)에서 始作될 때에는 8080 memory 構造로 되는 것이다.

한 segment 內의 byte 나 word 는 64 k byte segment 內에서 16-bit offset address 를 使用하여 address 된다. 1 mega byte

까지 addressing 하기 위하여 20-bit의 physical address 를 形成하여야 하는데 그것은 그림1 과 같이 16-bit offset address를 16-bit segment address에 加하되 segment address의 LSB의 右側에 모두 0의 값을 갖는 4 bit 를 이은 것을 使用한다.

Input /Output Space :

8080에서는 256 種의 I/O port 만이 可能하나 8086에는 64k 의 address 可能한 I/O port가 있다. Memory와는 달라서 I/O space 는 segment register를 사용하지 않는 形態로 되어 있다. Input/output의 實際의 address 는 20-bit로 되어 있지만 높은 쪽 4

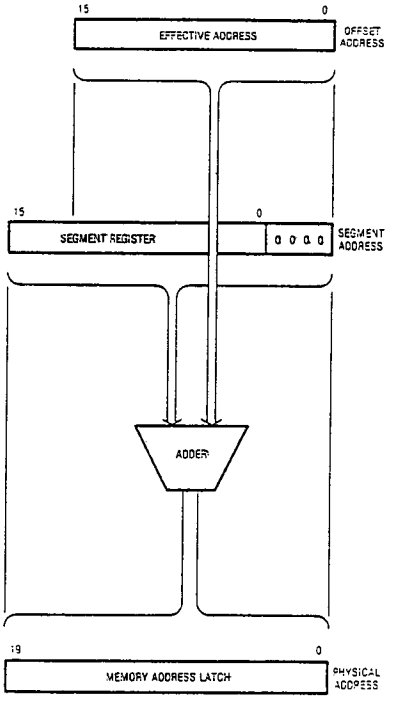

그림 1. 20-bit physical memory 의 形成

bit 는 항상 0의 값을 갖는다. 入出力 port 는 8-bit 또는 16-bit 길이로 使用할 수 있고 16-bit port는 홀수 또는 짝수 address를 使用할 수 있다.

b. Register 의 構成

8086 processor는 16-bit register 4개씩을 한 group으로 한 register file이 3개 組가 있고 9개의 1-bit flag가 한 組를 이루고 있다. 3個組의 register file은 general register file로 되어 있다. 그리고 이밖에도 프로그래머가 直接 呼出할 수는 없지만 16-bit로 된 instruction pointer가 있는데 이것은 control transfer 命令에 의하여 使用된다. 따라서 그림 2와 같이 8080 register set 은 8086 register set 의 一部分에 不過하다. 8080과 8086에서 서

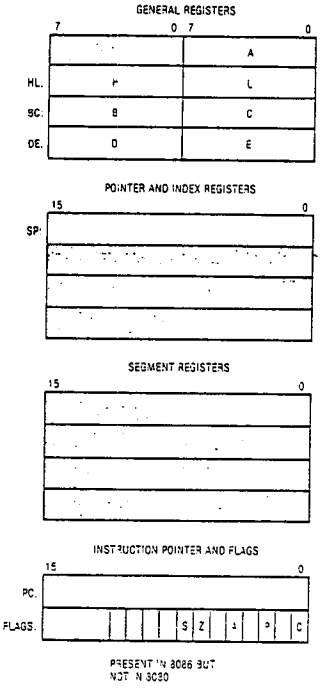

그림 2. 8080 및 8086 register set

로 對應되는 register라 할지라도 그림2와 그림3에서 보는 바와 같이 名稱은 서로 꼭 같지 않다.

General Register File :

그림 3의 AX-BX-CX-DX register

set 을 general register file 또는 HL group 이라고 부른다. 이 register 들은 8086의 數直演算 및 論理演算時에 交互的으로 使用된다. General register 들이 他 register 들과 다른 점은 高位 byte 가 따로 address할 수 있는 점

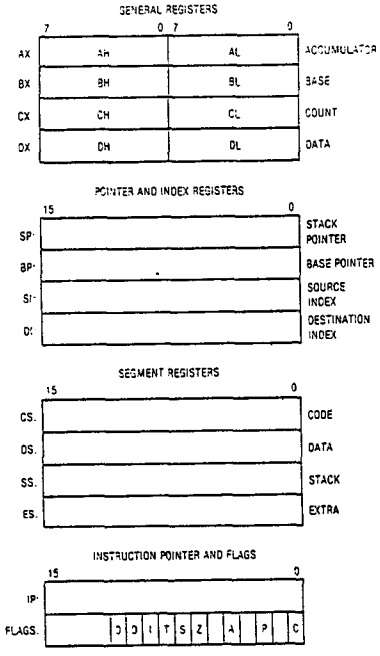

그림 **3.** 8086 register set

이다. 그러므로 general register 는 2個組 (H-file 및 L-file) 의 8-bit register group (4個의 register) 으로 形成되어 있다고도 생각할 수 있다.

Pointer 및 Index Register File :

그림 3의 SP-BP-SI-DI register set 은 pointer 및 index file 또는 P I group 이라고 부른다. 이 file 안에 있는 register 들은 대개 한 segment內에서 addressing 을 위하여 使用되는 offset address 를 包含하게 된다는 점에서 서로 類似하다. 이들도 general register 들과 같이 數直 및 論理演算에 使用된다. 이들은 모두 address 計算에 쓰이지만 pointer group (P-group) 과 index group (I-group) 으로 兩分된다.

Segment Register File :

그림 3의 CS-DS-SS-ES는 S-group 이라고 부른다. 이 segment register 들은 processor 의 memory addressing 에 重要한 役割을 한다. 이들은 모두가 memory address 計算에 使用된다. Mnemonic code에서 使用되는 각 segment rgister의 名稱은 그림 3에 表示한 것과 같다. CS register의 內容은 current code segment 를 定義한다. 모든 instruction fetch 에 있어서 instruction pointer (I P) 를 off-set 로 하여 CS 에 대하여 相對的으로 表示한다. DS register의 內容은 current data segment 를 定義한다. 一般的으로 BP나 SP를 包含하는 경우를 除外하고는 모든 data의 引用에 있어서는 DS를 基準으로 하여 相對的으로 나타낸다. SS register의 內容은 current stack segment 를 定義한다. SP나 BP를 包含하는 모든 data 의 引用에 있어서는 SS에 대하여 相對的으로 나타낸다. Call operation 이나 interrupt 및 return operation 에 의한 모든 push 나 pop operation 도 여기에 包含된다.

ES register 의 內容은 current extra segment 를 定義하며 이것은 또 하나의 data segment 로서 取扱되는 것 外에는 特別한 用途는 없다.

Segment register 들을 load 시키지 않거나 使用하지 않은 프로그램들을 "dynamically relocatable" 이라고 하며, 이와 같은 프로그램들은 interrupt 될 수 있고 memory 內에서 새로운 場所로 옮길 수 있으며 새로운 segment register 값을 가지고 다시 始作될 수 있다.

Flag Register File :

그림 3의 A-C-D-I-O-P-S-T-Z

register set은 flag register file 또는 F group이라고 부른다. 이 group內의 각 flag 들은 모두 1-bit로 되어 있고, processor의 狀態(status)에 관한 情報를 기억하였다가 processor를 制御하는 데 쓰인다. 여기에는 다음과 같은 mnemonic code가 使用된다.

AF : auxiliary cary	CF: carry
DF : direction	IF: interrupt enable
OF : overflow	PF: parity
SF : sign	TF: trap
ZF : zero	

이들 中에는 8080과 共通인 것도 있고 DF flag은 string의 取扱命令 (自動增加 또는 自動減少) 의 方向을 制御하며, IF flag은 外部로부터의 interrupt를 可能 또는 不可能하게 한다. TF flag은 프로그램 debugging을 하기 위하여 processor를 single-step mode로 動作하게 한다.

c. Instruction Set

8086의 instruction set은 8080의 大部分의 instruction을 包含하며 address operand 指定方法이 보다 多樣하고 모든 分野에 걸쳐 더 잘 되어 있다. 이것은 특히 Pascal 이나 Cobol 과 같은 block-structured language 프로그램을 能率的으로 實行할 수 있도록 되어 있다. 거의 모든 命令이 8-bit 나 16-bit 演算이 可能하다. Data transfer에는 4種이 있고 네 가지 算術演算이 모두 可能하다. 여기에는 새로운 論理命令인 test 도 包含되어 있다. 이밖에도 새로운 것은 byte 및 word형 文字의 處理와 segment 間 傳達이다.

Operand addressing :

이것은 2-演算子 (two-operands) 식 命令을 쓸 수가 있는 데, 하나의 register나 memory address를 하나의 演算子로 使用하고 또 다른 register나 常數를 다른 演算子로 쓸 수 있다. 一般的으로, memory operand 들은 16-bit offset address로 直接 address 되거나, base register (BX 또는 BP)나 index register (SI 또는 DI)의 內容을 8-bit 또는 16-bit displacement 常數에 더하여 間接的으로 address하기도 한다. Immediate 常數를 除外하고는 single-operand 演算도 可能하다. 實際로 모든 8086 演算은 8-bit 로도 할 수 있고 16-bit 로도 할 수 있다.

Memory operand :

Memory안에 있는 operand를 指定하는 데는 4가지 方法이 있다.

① 直接 16-bit offset address
② Base register를 通한 間接address
③ Index register를 通한 間接 address
④ Base register 와 index register의 合을 通한 間接 address

General register BX와 pointer register BP는 base register로 使用할 수 있다.

Register operand :

4개의 16-bit general register와 4개의 16-bit point 및 index register들은 16-bit에 있어서 交互的으로 演算子로 쓸 수 있다. 다만, 乘算, 除算, 및 文字處理에 있어서만은 例外로 AX register를 사용된다. 그러나 乘算, 除算, 및 文字處理등은 AL을 使用한다.

Immediate operand :

乘算, 除算, 및 文字處理의 경우를 除外하고는 모든 2-operand 演算에 있어서 하나의 source operand는 命令語內에 있어서 immediate data로 나타날 수 있다.

16-bit immediate operand의 높은 쪽 byte가 낮은 쪽 byte의 符號의 延長으로 되어 있

을 때에는 8-bit operand로 縮少시킬 수 있다.

Addressing mode :

8086의 addressing mode에 있어서는 BX 및 BP register를 base register로, SI 및 DI register를 index register로 使用할 수 있다. 例를 들면 ;

단순 變數와 配列의 경우 : 단순 變數들은 direct address mode로 呼出되고, 配列元素는 indirect address mode로 呼出되는데, register SI (SI에는 配列에 대한 index가 包含되어 있음)와 displacement (displacement에는 segment 內에서의 配列의 offset를 包含하고 있음)를 合한 것을 사용한다.

Based variable의 경우: based variable은 어떤 다른 變數에 의하여 指示된 memory address에 의하여 呼出된다. 만일 pointer variable의 內容이 BX에 기억되어 있다면 BX를 사용하는 indirect addressing mode에 의하여 based variable을 配列이고 配列에 대한 index가 SI에 기억되어 있다면 BX와 SI의 合에 의한 indirect addressing mode로서 그 配列의 元素들을 呼出하게 될 것이다.

Stack marker의 경우 : stack marker를 사용하므로써 block 構造의 言語를 處理하는데 있어서 더욱 能率的으로 할 수 있게 되어 있고 reentrant 節次를 위한 能率的인 address 方法이 마련되어 있다. register BP를 stack內에서 活動記錄(activation record)의 始作點을 가르키는 stack marker로 사용할 수 있다. Base register BP와 displacement의 合에 의한 間接 address mode를 써서 現在 사용되고 있는 block 內에서 宣言된 變數를 呼出할 수 있다.

Data transfer :

Data tranfer 方式에는 汎用 transfer, accumulator 指定 transfer, address-objective transfer, 및 flag transfer 등 4가지로 나눈다.

汎用 data transfer 操作은 move, push, pop exchange 등이다. 이러한 操作은 모든 種類의 operand에 대하여 할 수 있다. Accumulator 指定 transfer는 input, output, 및 translate operation을 포함한다. 이들의 처음 256 port는 8080에서와 같이 直接 address될 수 있다.

그러나 8086은 DX register를 通하여 間接으로 address되는 port도 있다. 後者의 方法으로는 64k의 port까지 address可能하다. 8080은 8-bit 길이의 port만 사용할 수 있지만 8086은 8-bit 또는 16-bit 길이의 port를 모두 사용할 수 있다.

Arithmetics :

8080은 符號를 갖지 않는 數의 演算은 8-bit의 加減算만 可能하였으나, 8086은 4가지 演算이 모두 可能하다. 8-bit 演算과 16-bit 演算이 모두 可能하고, 符號붙은 演算이나 붙지 않은 演算이 다같이 可能하다. 負數에 대해서는 標準 2's complement 表現方法을 쓰고 있다. OF flag을 사용하면 符號붙은 overflow 狀態도 檢出된다.

乘算 및 除算 :

乘算 및 除算은 符號가 붙었거나 붙지 않았거나 다 可能하다. 乘算의 結果는 길이가 2倍로 되고(8-bit 乘算時는 16-bit, 16-bit 乘算時는 32-bit로 된다.), 16-bit의 被除數를 8-bit의 除數로 나눌때에는 8-bit의 몫과 8-bit의 나머지를 얻게 된다.

Transfers :

8086의 call, jump, return 命令에는 두 가지 基本型이 있다. 즉, current code segment 內에서 control을 傳達하는 segment內 transfer가 있다. 뿐만 아니라 直接 및 間接 transfer가 모두 可能하지만 8080에서는 直接 segment

間 transfer 만이 可能하다.

Interrupt :

8086의 Interrupt 機構는 8080의 그것보다 훨씬 더 一般的이다. 8086 processor는 2가지 形態의 外部 interrupt (non－maskable interrupt 및 maskable interrupt)를 認識할 수 있으며 이를 識別하기 위한 pin이 한개 마련되어 있다.

Interrupt가 發生되면 새로운 code segment에 있는 새로운 場所로 control이 傳達된다. 256素子의 表(interrupt transfer vector)가 interrupt service code 場所에 대한 pointer를 包含하고 memory의 처음 부분에 기억된다. 각 素子는 4 byte 길이로 되어 있으며, offset address와 service code segment를 위한 segment address를 包含한다. 이 表의 각 素子는 interrupt type에 該當되며 이 type들은 0부터 255까지 番號가 붙여져 있다. 모든 interrupt는 flag 狀態를 stack에 push 하고 傳達을 行하며, interrupt transfer 벡터를 通하여 間接的인 call을 行한다.

프로그램 實行 control은 外部 interupt와 類似한 操作에 의하여 傳達될 수 있다. 어떠한 型(type)의 interrupt라도 發生시킬 수 있는 一般化된 2-byte 命令이 있는데 interrupt의 型은 두번째 byte에 의해서 指定된다. 그리고 하나의 特別한 型의 interrupt를 發生시키는 特別히 만들어진 1－byte 命令도 있다. 이와 같은 命令은 소프트웨어 debugging 時에 必要하다. interrupt가 끝났을 때에는 interrupt return 命令을 使用하면 subroutine return 機能을 遂行함과 同時에 기억시켰던 flag 狀態를 復歸(pop) 시킨다.

結 言

Intel 8086 microprocessor architecture는 廣範圍한 應用에 適合하도록 設計되어 있다. 8086의 對稱的 操作機構는 소프트웨어 시스템 특히 高水準 言語의 實現에 效果的으로 되어 있다. 8086의 數値演算, 論理演算, byte－string 및 bit 操作能力등은 매우 優秀하기 때문에 演算을 必要로 하는 各種 應用에 매우 適合하다. reentrant code, position－independent code, 및 dynamically relocatable program등을 뒷받침할 수 있는 能力이 있기 때문에 規模가 큰 應用 시스템에 있어서 合理的인 operating system 實現을 可能케 할 수 있을 것이다. 充分히 큰 記憶容量을 가지고 있기 때문에 이와 같은 目的에 잘 符合된다고 볼 수 있을 것이다. 實로 8086은 8080 processor의 發展된 形態로써 많은 脚光을 받고 있는 16－bit processor 中의 하나 임에는 疑心할 餘地가 없다고 하겠다.

參 考 文 獻

1. S.P. Morse, W.B. Pohlmen; B.W. Ravenel "The Intel 8086 Microprocessor : A 16－bit Evolution of the 8080", IEEE Computer, June 1978.

◇