16-Bit 마이크로프로세서 時代의 主役들 : 〔2〕 Intel 8086

金 悳 鎭

高麗大學校 工科大學 電子工學科 敎授(工博)

1. M 68000 Microprocessor 開發의 必要性

지난 數年間 microprocessor 技術은 繼續 發展을 거듭하여 왔다. 그 중에서도 注目할 만한 것은 單一 칩 microprocessor 上에 集積할 수 있는 回路의 密度가 더욱 높아졌다는 事實이다. 從前의 microprocessor 들은 普通 칩當 5,000 ~ 10,000 個의 트랜지스터를 包含하였으나 最近의 프로세서들은 25,000~70,000 個의 트랜지스터를 集積할 수 있게 되었다. 發展된 것은 回路의 集積密度 뿐만 아니라 回路의 動作速度와 電力損失도 그에 相應하여 改善되었다. 이러한 發展은 技術의 進步의 結果라고 보아야 할 것이다. 集積의 高密度化는 結局 回路의 dimension 을 더욱 縮小시킬 수 있는 wafer 處理技術에 基因된다고 볼 수 있다. Wafer 處理技術의 進步는 앞으로도 繼續될 것이며 이에 알맞는 새로운 microprocessor 構造도 登場하게 될 것이다.

a) 市場性

數年前만 하더라도 microprocessor 生産業者들은 今日과 같은 需要를 豫想하지 못하였었다. 初期의 microprocessor 設計者들이 그들의 製品이 廣範圍하게 使用될 것을 豫測하지 못하였던 것처럼 今日의 設計者들도 새로운 microprocessor 의 終局的인 應用以上의 것은 想像하기가 어려울 것이다. 다만 오늘의 設計들이 주는 暗示는 앞으로의 設計는 보다 더 廣範圍한 應用을 可能하게 하기 위하여 一般的이고 柔軟性있게 設計되어야 한다는 點이다.

b) 高價의 software

Microprocessor 應用에 있어서의 software 費用 問題는 中小型 컴퓨터의 경우보다 더 甚刻하다. 계속되는 메모리 價格의 低下, processor 機能의 向上 및 利用方法의 複雜多樣化 等으로 말미암아 microprocessor 프로그램은 점점 더 複雜해지고 또 規模도 커져 가고 있다. Hardware 의 價格이 數 100 弗 밖에 되지 않는 microprocessor 의 software 價格이 때로는 10 萬弗 또는 그 以上이 될 때도 있다. Microprocessor 의 hardware와 software 의 이와 같은 價格의 不均衡은 大量生産時에는 問題가 안되지마는 少量需要時에는 生産을 困難하게 한다. 그러므로 software 價格을 低下시키기 위하여 microprocessor 設計者들은 高水準 言語의 使用을 容易하게 하여야 하고 잘 開發된 프로그래밍 技法을 提供하여 주어야 할 것이다.

c) 높은 設計費用

數萬個의 트랜지스터를 集積하여 새로운 microprocessor를 設計하고 製造하는 데에는 대단히 많은 費用이 든다. 이 設計에는 勿論 CAD(computer-aided design)를 하여야 하는데 그 費用이 또한 적지 않게 든다. 設計費用을 줄이는 方法은 여러 가지 있을 수 있다. 첫째는, 一般的인 構造로 바로 設計하는 것으로써, 이 方法은 實現하고, 試驗하고 修正하기가 쉽고 다른 方法에 比해 費用이 덜 든다. 둘째는, 可能한 限 오래 持續될 수 있고, 將次 쉽게 擴張시킬 수 있는 새로운 構造로 設計하는 것이다. Processor 메이커들은 몇 年마다 한번씩 새로운 構造의 製品을 生産하기는 어렵기 때문이다. 現在의 8-bit microprocessor 構造를 擴張시키고 改良하는 데에서 얻은 經驗을 土臺로 하여 앞으로 보다 더 擴張, 改良할 수 있도록 되었다. 設計者들은 장차 豫想되는 擴張可能性에 對하여 制限을 두지 않도록 하여야 한다. 이들이 經驗한 지난 날의 가장 큰 잘못은 address 의 크기를 制限하였던 點과, 장차 必要하게 될런지도 모르는 새로운 命令을 쓸 수 있게 하기 위하여 使用하지 않는 operation code를 준비해 두지 못했던 點이다.

d) MC 68000 의 設計目標

Motorola 68000 microprocessor의 構造는 上述한 要求條件들을 滿足시키도록 設計되었다. 이 processor 의 規格을 표 1에 나타내고 있다.

표 1. Motorola 68000의 規格

Integer sizes	8, 16 및 32 bits
Addressing capability	16, 777, 216 bytes
Input／output	memory-mapped
Technology	HMOS
Internal cycle	250ns
Memory access	500ns
Relative performance	MC 68000의10～25 倍
Pin 數	64
Power	+5V

68000의 設計目標는 다음과 같다.

i) 68000의 設計는 여러 가지 다른 version (implementation)을 가질 수 있는 'family' 컴퓨터 構造를 가지고 있다. 그 中 最初의 version인 MC 68000은 現在의 技術的 制限이 許容하는 完全한 68000 'family' 構造中 一部를 實現시킨 것이다.

ii) 68000 family는 프로그램하기 쉽게 設計되어 있다. 이 processor에는 不自然스런 制限이라든가, 構造上 不便한 것들이 可能한 限 많이 制外되었다.

iii) 68000 family는 汎用構造를 가지고 있기 때문에 汎用 microprocessor의 增大되는 市場性을 充足시켜 줄 수 있다.

iv) 68000 family는 浮動小數點演算, 文字處理 等이 容易하도록 되어 있다. 使用하지 않는 空間은 장치 必要에 따라 새로운 回路를 追加할 수 있도록 하였다.

v) 68000 family는 高水準 言語를 實行하는 데 適合하도록 되어 있으며, Motorola는 잘 알려진 高水準 言語로 開發된 software를 供給할 수 있다.

2. 68000의 內部構造

a) Resources

68000 컴퓨터는 2^{32} bytes (初期의 것은 2^{24} bytes로 制限했었음.)의 address space를 가지고 있다. 메모리는 'byte'로 address되나 bit 操作命令에 對해서는 個別的인 bit의 addressing도 可能하다. 메모리는 1, 8, 16 또는 32 bit 單位로 呼出 可能하다. CPU resource로서는 16個의 32-bit register, 1個의 32-bit program counter (初期에는 24 bits) 및 1個의 16-bit status register로 構成되어 있다.

그림 1에 보는 바와 같이 register들은 크게 2 class로 나뉘어진다. 데이터 取扱을 위해서는 1次的으로 8個로된 data register가 使用된다. 이 register들은 모든 演算에 있어서 演算子(operand)로 使用되기도 하고 結果를 저장시킬 register로도 使用되는데 addressing에 있어서는 index register로만 使用된다. 남은 8個의 register인 address register들은 主로 addressing에 使用된다. 이밖에도

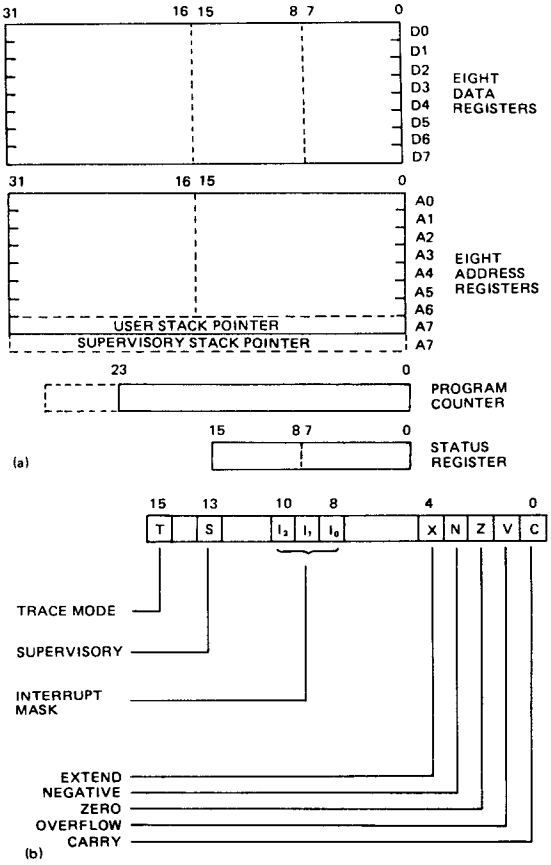


그림 1. (a) MC 68000의 프로그래밍 모델 및
(b) Status register의 內部構造

program counter와 status register가 別途로 마련되어 있다.

b) Addressing

메모리는 8-bit byte, 16-bit word 또는 32-bit long word 길이로 address된다.

Address 計算은 표 2에 나타난 바와 같이 命令語의 6-bit field로 指定한다. Addressing을 必要로 하는 어떤 命令語에 있어서도 모든 addressing mode를 使用할 수 있다.

Address는 32 bit 單位(現在에는 24 bit)로 되어 있다. 이와 같은 構造는 2^{16} byte 보다 address space가 적은 小型 시스템에 對해서도 效果的으로 對處할 수 있게 되어 있다. 例를 들면 한 命令語에 含된 絶對 어드레스(absolute address)는 16 bit 또

표 2. MC 68000 addressing mode

REGISTER DIRECT ADDRESSING:	
data register direct	EA = Dn
address register direct	EA = An
status register direct	EA = SR
REGISTER DEFERRED ADDRESSING:	
register deferred	EA = (An)
register deferred post-increment	EA = (An): An < – An + N
register deferred pre-decrement	An < – An – N: EA = (An)
base relative	EA = (An) + d16
indexed	EA = (An) + (Xn) + d8
PROGRAM COUNTER RELATIVE:	
relative with offset	EA = (PC) + d16
relative indexed	EA = (PC) + (Xn) + d8
short PC relative branch	EA = (PC) + d8
long PC relative branch	EA = (PC) + d16
ABSOLUTE ADDRESSING:	
absolute short	EA = (next instruction word)
absolute long	EA = (next two instruction words)
IMMEDIATE DATA ADDRESSING:	
immediate	DATA = next instruction word(s)
quick immediate	DATA = Subfield of instruction (4 bits)

DEFINITIONS:
EA = effective address
An = address register
Dn = data register
Xn = address or data register used as index register
SR = status register
PC = program counter
d8 = 8-bit displacement
d16 = 16-bit displacement
N = 1 for byte, 2 for word, and 4 for long word operands
( ) = contents of
< – = replaces

는 32 bit register를, 그리고 index 計算에 있어서는 16 bit 또는 32 bit register를 入力으로 使用할 수 있다. 이와 같은 特性을 가지고 있기 때문에 少數의 address 만을 必要로 하는 대단히 큰 address를 프로그램의 效率을 低下시킴이 없이 뒷받침 할 수 있다. Address의 크기를 16 bit로 할 것인지 32 bit로 할 것인가는 각각 個別的으로 指定하게 되어 있기 때문에 한 프로그램 內에 이들 크고 작은 address를 混用할 수 있다.

68000에서 使用할 수 있는 addressing mode들은 다음과 같다.

Register direct addressing : data register나 address register가 演算子를 包含한다.

Address register deferred addressing : 演算子 어드레스(operand address)가 指定된 address register에 나타난다.

Address register deferred post-increment addressing : 演算子 어드레스는 指定된 address register에 나타난다. 이 演算子가 呼出된 뒤에 그 register의 address가 演算子의 크기(1, 2 또는 4) 만큼 增加된다.

Address register deferred pre-decrement addressing : 演算子 어드레스는 指定된 어드레스 register에 나타난다. 演算子가 呼出되기 前에 address register의 內容이 演算子의 크기 만큼 減少된다.

Base relative addressing : 演算子 어드레스는 指定된 address register의 內容과 命令語內에 있는 16-bit 符號가진 變位値(displacement)와의 合이 가르키는 값이다.

Program counter relative : 演算子 어드레스는 program counter의 現在의 內容에다 命令語內에 있는 16-bit 符號가진 變化値를 합친 값이다.

Indexed addressing : 演算子의 어드레스는 指定된 address register의 內容과 또 하나의 register(data register 또는 address register)의 內容과 命令語內에서 指定한 8-bit 符號가진 變化値와의 合으로 決定된다.

Program counter indexed addressing : 演算子의 어드레스는 program counter의 現在의 값과, 指定된 data register 또는 address register의 內容과 命令語內에 있는 8-bit 符號가진 變化値와의 合으로 決定된다.

Absolute addressing : 演算子의 어드레스는 命令語에 그대로 包含된다.

Immediate addressing : 演算子 그 自體가 命令語內에 包含된다.

Bit addressing : Bit 取扱命令으로서는 set, change, clear, test 등이 있다. 이러한 命令을 使用할 때에는 各 memory word를 上述한 addressing mode로 addressing 한다. 그리고 處理될 各 bit는 命令語內의 bit 番號에 依하여 address 된다. 이렇게 하므로써 bit 選擇을 위한 論理의 命令語나 마스크 作用을 하는 프로그램 없이 간단히 各 bit를 address 할 수 있게 된다. 各 register들에 對해서 32 bit 모두 個別的으로 address 할 수 있다.

c) Data의 取扱

그림 2에 나타낸 바와 같이 68000은 여러 가지의 data 型式에 對하여 完全한 演算을 할 수 있다. 一般的으로 addressing mode는 data 型式과 無關하다. 또 演算子의 size도 演算의 種類와 關係없이 指定될 수 있다. 그리고 operand의 源泉(source)은 register가 될 수도 있고 address된 memory가 될 수도 있다. 演算結果는 register나 指定된 memory에 저장된다. 이러한 種類의 "register-to-memory" 演算은 結果를 기억시키기 위한 register 數를 節約할 수 있게 한다. 大部分의 演算에 있어서 memory-to-register, register-to-register, immediate-to-register, 또는 immediate-to-memory 등의 演算을 指定할 수 있다. 'Move 命令'은 全

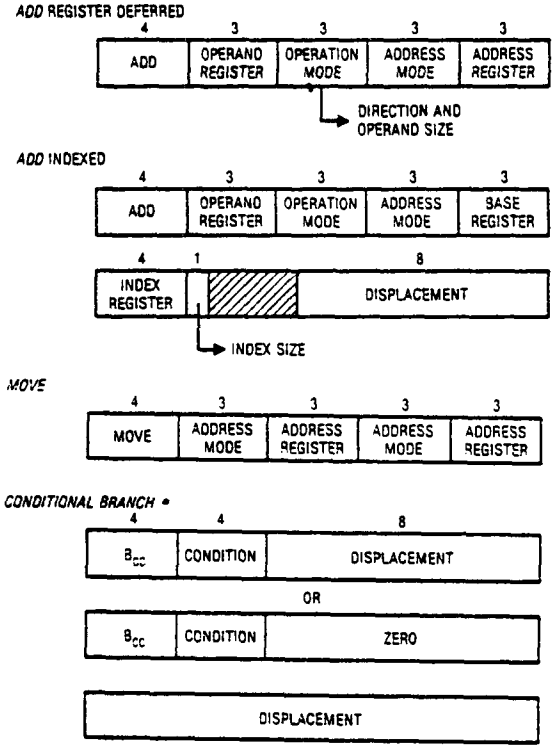


그림 2. MC 68000 命令型式

2-address 命令型式을 가지며 더욱 많은 융통성을 가지고 있다. 이 命令으로는 memory-to-memory 傳達演算도 할 수 있다.

68000의 data型式과 이들의 演算의 種類는 다음과 같다.

Integer 演算 : 整數演算은 add, subtract, multiply, divide, negative, compare 및 arithmetic shift 等이 可能하다. 整數의 길이는 1 byte, 2 byte 및 4 byte 中 任意로 選擇할 수 있다. Shift 動作은 left shift 이거나 right shift 이거나 다 같이 multiple-bit shift가 可能한데, 이것은 命令語內에 shift 回數를 指定하는 方法도 있고, data register內에서 미리 shift 回數를 計算하는 方法도 있으며 overflow가 發生 하면 그 狀態가 表示된다.

Multiprecision integer 演算 : Add with extend, subtract with extend, negative with extend, unsigned multiply 및 unsigned divide 等의 multi precision integer 演算을 할 수 있다. 乘算과 除算은 2-byte 演算子만을 使用하며, 그 밖의 演算은 1, 2, 4 byte 演算이 모두 可能하다.

Logical 演算 : AND, OR, exclusive OR, complement, compare, shift 및 rotate 演算이 可能하다. 이 演算은 1, 2, 4 byte로 할 수 있다.

Bool 演算 : AND, OR, exclusive OR, complement, implication, set according to condition code 等의 Bool 演算이 可能하다.

Bit 演算 : Set, clear, change, test 等의 bit 演算이 可能하며 bit는 獨立的으로 address 할 수 있다.

Decimal 演算 : Add, subtract, negate, compare 演算이 可能하다. BCD 命令은 메모리에 있는 數字들(memory-to-memory) 한번에 2個의 數字(1 byte)씩 演算한다. 可變長 memory-to-memory 10進 演算을 할 때에는 loop命令과 함께 使用하면 된다.

Character 演算 : Move와 compare 命令을 메모리에 있는 演算子에 對하여 實行할 수 있다.

Address 演算 : Increment, decrement, add integer, subtract integer, compare 및 load effective address 等의 演算이 可能하다.

Real 演算 : Floating-point 型式의 add, subtract, MC 68000 컴퓨터의 命令 set은 표 3과 같다.

표 3. MC 68000 命令 set

Mnemonic	Description
ABCD	Add decimal with extend
ADD	Add
ADDX	Add with extend
AND	Logical and
ASL	Arithmetic shift left
ASR	Arithmetic shift right
BCC	Branch conditionally
BCHG	Bit test and change
BCLR	Bit test and clear
BRA	Branch always
BSET	Bit test and set
BSR	Branch to subroutine
BTST	Bit test
CHK	Check register against bounds
CLR	Clear operand
CMP	Arithmetic compare
DCNT	Decrement and branch non-zero
DIVS	Signed divide
DIVU	Unsigned divide

Mnemonic	Description
EOR	Exclusive or
EXG	Exchange registers
EXT	Signed extend
JMP	Jump
JSR	Jump to subroutine
LDM	Load multiple registers
LDQ	Load register quick
LEA	Load effective address
LINK	Link stack
LSL	Logical shift left
LSR	Logical shift right
MOVE	Move
MULS	Signed multiply
MULU	Unsigned multipiy
NBCD	Negate decimal with extend
NEG	Two's complement
NEGX	Two's complement with extend
NOP	No operation
NOT	One's complement
OR	Logical or
PEA	Push effective address
RESET	Reset external devices
ROTL	Rotate left without extend
ROTR	Rotate right without extend
ROTXL	Rotate left with extend
ROTXR	Rotate right with extend
RTR	Return and restore
RTS	Return from subroutine
SBCD	Subtract decimal from extend
SCC	Set conditionally
STM	Store multiple registers
STOP	Stop
SUB	Subtract
SUBX	Subtract with extend
SWAP	Swap data register halves
TAS	Test and set operand
TRAP	Trap
TRAPV	Trap on overflow
TST	Test
UNLK	Unlink stack

multiply, divide 等이 可能하다.

String 演算 : String move, string search, tr translte 等의 string 演算이 可能하다.

d) 프로그램 制御

프로그램 制御 命令으로는 conditional branch, jump, jump to subroutine, return from subroutine 및 return from interrupt 가 있다. TRAP 命令을 使用하면 16 가지의 다른 operating-system call 을 指定할 수 있고, STOP 命令은 processor 를 멈추며, RESET 命令은 시스템을 리세트 시켜주며, MOVE 命令은 processor의 status word를 다룰 수 있다.

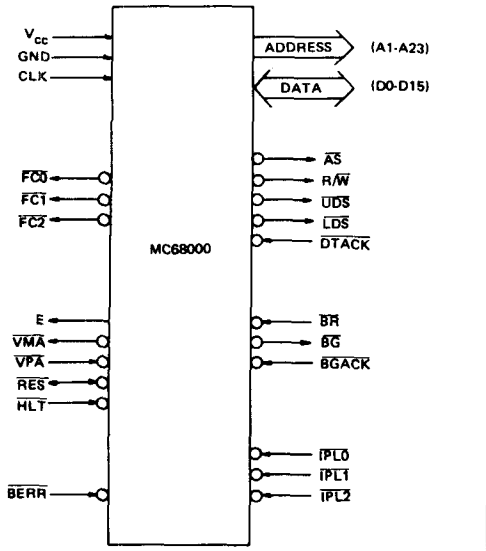
3. 6800 시스템 構造

여기에서 시스템 構造라함은 processor와 그의 周邊裝置와의 interaction에 關한 것으로서 interrupt 構造, memory segmentation, bus interface, 入出力 構造 等을 말한다.

68000 system의 構造는 可能한 많은 융통성을 갖도록 設計되어 있다. 例를 들면, I/O device register 들은 다른 Motorola microprocessor 와 마찬가지로 메모리番地(memory-mapped)로 番地를 指定한다. Memory mapped I/O 構造에 依하여 프로그래머에게 많은 융통성을 賦與하게 되며 device 制御와 data register 를 다루는데에 있어서 모든 命令 set 를 效果的으로 活用할 수 있는 能力을 갖게 할 수 있다. 入出力裝置制御에 있어 別途의 命令을 必要로 하지 않으므로 processor 는 더욱 簡單해지고 命令 set 는 기억하기가 쉽다. I/O 空間은 메모리의 critical area 를 保護하는데 使用하는 memory-management 方法을 使用하여 保護한다.

68000의 bus 構造도 또한 간단하고, 빠르고, 융통성 있게 設計되어 있다. Address bus 와 data bus 는 分離되어 있어서 multiplexing 이 必要없다. 이렇게 하므로써 別途의 demultiplexing 裝置가 必要하지 않으며, 따라서 處理速度가 特히 問題되는 시스템에 있어서 最大의 性能을 發揮할 수 있게 되어 있다. 이 bus 는 非同期式이므로 여기에 data 를 傳送할 때에는 handshake 信號를 使用한다. 이 band-shake 信號를 使用하므로써 入出力裝置나 기억장치의 應答時間이 多樣한 形態의 것들까지도 같은 processor bus 에 連結할 수 있다

간단한 bus request/grant protocol 을 使用해서 processor 와 DMA (direct memory access) 裝置는 別途의 論理裝置 없이도 system bus를 共有



A1-A23	Address leads	23-bit address bus: capable of addressing 16,777,216 bytes in conjunction with UDS and LDS.
D0-D15	Data leads	16-bit data bus: transfers 8 or 16 bits of information.
AS	Address strobe	Indicates valid address and provides a bus lock for indivisible operations.
R/W	Read/write	Defines bus operation as read or write and controls external bus buffers.
UDS, LDS	Data Strobes	Identifies the byte(s) to be operated on according to R/W and AS.
DTACK	Data transfer acknowledge	Allows the bus cycle to synchronize with slow devices or memories.
BR	Bus request	Input to the processor from a device requesting the bus.
BG	Bus grant	Output from the processor granting bus arbitration.
BGACK	Bus grant acknowledge	Confirmation signal from BG indicating a valid selection from the arbitration process.
IPL0, IPL1, IPL2	Interrupt priority level	Provides the oriority level at the interruating tunction to the processor.
FC0, FC1, FC2	Function code	Provices external devices with information about the current bus cycle and processor state.
CLK	Clock	Master TTL input clock to the processor.
RES	Reset	Provides reset (initialization) signal to the processor and peripheral devices.
HLT	Halt	Stops the processor and allows single stepping.
BERR	Bus error	Provides termination of a bus cycle if no response or an invalid response is received.
E	Enable	Enable clock for 6800 systems.
VPA	Valid peripheral address	Identifies addressed area as a 6800 compatible area.
VMA	Valid memory access	Indicates to 6800 family devices that a valid address is on the bus.
V_{cc}	-5 volt (2 pins)	–
GND	Ground (2 pins)	–

그림 3. MC 68000 CPU 칩

할 수 있다. 68000 칩에는 memory access를 잘못하였을 때, 어떠한 位直에서라도 命令의 實行을 中斷하고 trap이 일어나도록 하는 bus-falt input pin이 마련되어 있다. 이렇게 해서 memory가 保護될 수 있게 되어 있다.

68000의 interrupt 構造는 대개의 minicomputer에서 쓰고 있는 것과 비슷하다. 即 8個의 priority level을 가지고 있고, software가 interrupt-handling routine을 充分히 制御하고 實行할 수 있도록 interrupt vector 시스템으로 構成되어 있다. Processor의 現 priority level이 status word에 나타나며, 現 priority level 보다 낮거나 이와 같은 interrupt는 禁止되어 진다. 그러나 現 레벨보다 priority가 높은 것이 들어오면 interrupt가 일어난다. Interrupt가 받아 들여지면 processor는 'acknowledge' 信號를 내 보내며, interrupt를 願하는 裝直는 vector 番號로 應答한다. 그리고 이 vector 番號는 processor에 依하여 interrupt hadler의 入口를 찾기 위하여 메모리內에서 interrupt vector의 table을 index 하기 위하여 使用된다.

4. 68000 칩 設計

單一칩 MC 68000은 68000 architecture의 部分的인 實現이며 그의 핀 接續을 그림 3에 보이고 있다. Address 핀은 23個이며 23 bit address bus에 連結되고 16,777,216 byte 까지 address 할 수 있다. 데이터·핀은 모두 16個로써 16-Bit data bus에 接續되어 8 bit 또는 16 bit data를 傳達한다. 32-bit data인 경우에는 2번 memory로부터 呼出하여야 한다. 이 칩의 핀數는 모두 64個이다. 回路의 集積密度의 限界로 말미암아 實現시킬 수 있는 命令數도 制限을 받고 있다. 그 때문에 命令 코오드 中 $\frac{1}{8}$은 現在까지 實現시키지 않고 있다. 集積回路技術의 發展과 더불어 不遠間 意圖가 完全한 architecture가 實現될 날이 올 것이다.

參 考 文 獻

1. E. stritter and T. Gunter, "A Microprocessor Architecture for a Changing World : The Motorola 68000" IEEE, Computer, February 1979
2. B. L. Peuto and L. J. Shustek, "Current Issues in the Architecture of Microprocessors," Computer, Vol. 10, No. 2, Feb. 1977, PP. 20-25.
3. D. R. Allison, "A Design Phylosophy for Microcompter Architectures," Computer, Vol. 10, No. 2, Feb. 1977, PP. 35-41.