講座

16-Bit 마이크로프로세서 時代의 主役들 ; 〔3〕 Z 8000

金 惠 鎭
高麗大學校 工科大學 電子工學科 敎授(工博)

1. Z 8000 構造의 開発目標

Zilog 社에서 새로운 마이크로프로세서를 開發하게 된 것은 價格/性能·比를 잘 維持하면서 制御 및 處理能力을 크게 改善하기 위해서였다. 實際로, 그 동안 蓄積된 技術을 바탕으로 프로세서의 能力을 相當히 向上시킬 수 있게 된 것이다. 그러나 새로운 프로세서에 對한 더 重要한 期待는 一般性을 갖게 하는 問題이다. Zilog 社는 하아드웨어와 소프트웨어에 對한 使用者들의 그 동안의 投資를 保護하면서 機能을 强化시킨 새로운 콤포넨트型의 시스템 構造를 開發하게 되었다. Z 8000 family는 從來 여러 개의 콤포넨트를 混合하여 實現되었던 일들을 單一 構造로 實現시킬 수 있게 만들었다.

Z 8000의 開發目標는 다음의 3가지로 集約될 수 있다. 즉, 첫째는, 能力의 向上(increased capability)이고, 둘째는 廣範圍한 處理能力을 가지면서 構造的인 調和性을 갖게 하는 것이고, 셋째는 明確한 인터페이스(interface)에 두고 있다.

이 3가지 目標를 具體的으로 살펴보면 다음과 같다.

i) 能力의 向上

現在의 8 bit 마이크로프로세서나 大部分의 16 bit 미니컴퓨터들은 記憶容量이 적은 것이 問題가 되고 있다. 그러기 때문에 Z 8000의 開發目標中의 하나는 8 M byte 나 되는 큰 記憶容量을 갖게 하는 것이었다. 또 하나의 目標는 더욱 많은 "resource"를 갖는 마이크로컴퓨터를 만드는 것으로서, 具體的으로 그 結果는16 個의 汎用 bit register를 갖게 되었고, data 型式은 數 bit의 것으로부터 32 bit의 型式을 가질 수 있으며, 現存 프로세서에는 없는 새로운 命令들, 例를 들면 "乘算" "除算" 等의 命令들과 컴파일러를 뒷바침하기 위한 特別한 命令들이 새로 包含되었다. 複雜한 問題에 對한 프로세서 應用을 容易하게 하려면 task switching, interrupt, trap, 두 가지 實行 mode 等 좋은 hardware 構成으로 multipro-gramming을 可能하게 하여야 된다. 그리고 좋은 operating system을 만들려면 좋은 hardware 保護시스템이 있어야 한다. 마이크로프로세서의 全體的인 시스템 性能을 向上시키기 위하여 Z 8000에는 메모리와의 사이에 16 bit data 通路를 마련하였다.

ii) 構造的인 並存性

지금까지의 마이크로컴퓨터 시스템 設計의 經驗에 비추어 보면 처음부터 새로운 family 構造를 設計하는 일은 드문 일이며, 대개는 規模가 작은 處理能力으로부터 規模가 큰 處理能力에 이르기까지 廣範圍한 處理能力을 갖도록 統一된 構造를 갖도록 旣存 시스템을 改良해 나가는 것이다. 어떤 한 family 構造를 가진 시스템의 需要가 小規模시스템에서 大規模시스템에 이르기까지 增加될 수 있다면 그 시스템은 오랫동안 存續하게 될 것이다.

Z 8000은 40핀 unsegmented version과 48핀 segmented version의 두 가지 型이 있으며, 이 들은 上述한 目標를 達成하도록 設計되어 있다. 이것은 小規模시스템 應用의 경우에는 하나 또는 둘 以上의 64 K byte 어드레스 領域을 가진 unsegmented version으로 使用될 수 있고, 中規模의 應用에 있어서는 segmented Z8000과 하나의 MMU(memory management unit)로 시스템을 構成하면 4 M byte의 어드레스 領域을 直接 呼出할 수 있게 된다. 그리고 大規模의 應用 시스템의 경우에는, segmented Z 8000과 여러 개의 M.M U를 使用하므로써 8 M byte 어드레스 領域을 數個까지 使用할 수 있다. Segmented Z 8000은 unsegmented mode로도 動作시킬 수 있으므로 兩시스템은 並存性을 갖고 있다. Z 8000은 hardware 를 여러 개를 組合하여 使用하면 multiprocessing 과 distributed processing을 할 수 있게 되어 있다.

iii) 明確性

마이크로프로세서 構造에 있어서 明確性은 Key interface가 얼마나 잘 定義되고 잘 指定되었는가를 나타내는 尺度가 된다. 이것은 把握하기가 좀 어렵긴 하지만어떤 새로운 콤포넌트가 附加될는지 알 수 없는 family 構造에 있어서는 重要한 問題가 아닐 수 없다. 마이크로프로세서의 基本的 構造에 對한 明確性이란 命令 set의 規則性과 擴張可能性, 그리고 operating system interface의 一般的이고도 단순한 取扱可能性 與否를 意味한다. 한편, 시스템 構造上의 明確性이란 여러 가지 콤포넌트들 사이에 있어서의 잘 定義된 通信方法이 있는가를 意味하기로 한다. Z 8000에 있어서 이들 콤포넌트들 間의 重要한 連結方法은 共通 시스템버스인 Z-bus 이다.

2. 基本 構造

i) Address Spaces

마이크로프로세서에 있어서는 하나 以上의 address space가 있고 各 address space는 可能한 限 큰 것이 바람직 하다. Z 8000에 있어서는 메모리와 I/O 장치는 各各 다른 address space로 區分된다. 메모리 reference는 命令이나 데이터의 呼出 또는 stack 呼出의 경우에 使用된다. Z 8000은 status line의 여러 가지 組合에 依하여 이들 中 어느 것을 呼出하는 것인지 指定된다. Address space를 여러 개로 分離하므로써 呼出할 수 있는 全體의 byte數를 增加시킬 수 있고 또 메모리를 保護하는 데에도 效果的이다. 各 address space의 크기는 使用된 Z 8000의 version에 따라 다르다. 卽, 40핀 package version은 各 address space가 最大 64 K byte까지 可能하고, 48핀 package version을 各 address space가 最大 8Mbyte까지 可能하다. 40핀 version은 프로그램이 작아도 되고 데이터 space도 큰 것이 必要하지 않은 시스템에 많이 使用된다. 이 경우 다른 address space를 使用하므로써 4 × 64 K byte까지 쉽게 address할 수 있다. 그리고 各 address space들을 hardware로 分離하게 되면 메모리 保護도 簡單히 이루어진다.

48핀 version을 하나 또는 그 以上의 MMU와 함께 構成하면, 메모리 再配置가 必要하고 보다 더 메모리 保護를 잘 해야하는 中規模 내지 大規模시스템에 利用할 수 있게 된다. 이 경우에는, 여러 개의 MMU를 使用하여 分離한 address space들을 "status" 狀態에 따라 區別할 수 있게 된다.

위에서 말한 어느 version을 使用하던지 이에는 關係없이 모두 各 address space에는 直接呼出(direct access)이 可能하다. 卽, 命令이나 레지스터에 나타나는 address 表示에는 實際의 모든 address space를 指定하는 데에 充分한 bit를 配定하고 있다.

ii) Registers

Z 8000은 基本的으로 memory-to-register 構造로 되어 있다. 그렇다고해서 全的으로 이에만 依存하고 있는 것은 아니다. 例를 들면 文字 데이터의 경우에는 memory-to-memory operation이 實行될 수 있고 節次와 處理가 바뀔 때에는 stack operation이 實行되기도 한다. Register access가 現在 많이 實現되고 있는 memory access보다 훨씬 빠른 速度로 이루어질 수 있으므로 register access 構造를 擇하면 性能의 많은 向上을 가져온다.

지금까지의 register-oriented machine에서의 經驗에 依하면 단 4個의 汎用 register만으로는 不足하며 그의 適定數는 8～32사이에 있다고 볼 수 있다. Z 8000은 byte, word(16 bit), long word(32 bit), 때로는 4倍長語(64bit)의 데이터를 取扱할 수 있다. 그림 1에서 보인 바와 같이, 데이터 길이를 16 bit로 할 때에는 register가 16個(RO～R 15)로 되지만, 32 bit register로 使用하면 8個(RRO～RR 14)로 되

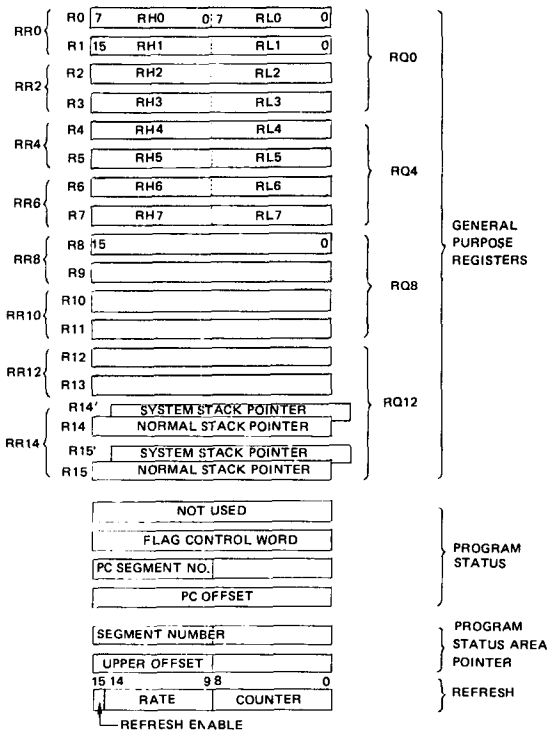

그림 1. CPU registers(segmented version)

고, 64 bit register로 쓰면 4個(RQ0～RQ12)로 된다. Address를 32 bit로 表示하므로 적어도 8個의 32 bit register가 必要하게 된다.

多小 制限은 있지만은 어느 register나 任意의 命令文에 依하여 累算器, source operand, 指標, 메모리 포인터 等으로 쓸 수 있다.

대개의 프로세서 應用에 있어서 몇 개의 register 들이 特定機能을 갖게 한다. 例를 들면, 대개의 高水準 言語는 하나의 stack pointer와 stack frame pointer를 必要로 한다. Z 8000과 같이 汎用 register만을 가진 시스템에 있어서는 使用者는 어떠한 制限도 받지 않고 이 들은 特定 機能을 賦與하여 使用할 수 있다. 마이크로프로세서를 어떻게 應用하게 될지 豫見할 수 없는 狀況아래에서는 위와 같이 register를 使用하는 것이 便利하다. Z 8000은 어느 register를 가지고도 stack으로 쓸 수 있다. Z 8000에는 2個의 hardware로 stack도 있으나 이 register들도 또한 汎用 register들이며 어떠한 演算에도 活用될 수 있다.

iii) Data Types

데이터 形式은 hardware的으로도 갖추어져야 되고 또 이에 適用되는 命令(instruction)도 있어야 된다. 새로운 데이터形式은 基本 데이터 形式에 依하여 simulate될 수 있으나 hardware的으로 實現되는 것이 演算을 더 빠르게 하고 便利하게 한다. 따라서 全的으로 hardware的으로 여러 가지 데이터形式을 實現하려면 컴퓨터 構造가 너무 複雜해지는 問題가 있다.

Z 8000은 몇 가지 基本的인 데이터 形式을 構造的으로 實現하고 이를 利用하여 擴張해 나갈 수 있다. 基本 데이터形式은 가장 頻繁히 使用되는 것들이다. 그리고 擴大된 데이터形式은 基本데이터形式을 使用하여 形成하는 것이며 現存하는 命令으로 다루게 되어 있다.

Z 8000에서는 基本 데이터形式은 "byte"이며 이것이 基本 呼出要素가 된다. 모든 다른 데이터 形式은 그들은 첫번째 byte의 address와 데이터의 길이(byte로 表示)로 나타낸다. Z 8000 構造는 이 밖에 또 다음과 같은 데이터 形式들도 使用할 수 있게 되어 왔다. 卽, byte(8bits), word(16bits), long word(32 bits), word string, 뿐만 아니라 한 byte나 word 內에 있는 各 bit들도 處理할 수 있다. BCD數는 2個의 4-bit數를 1byte로 나타내어 取扱한다. 위에 列擧한 모든 데이터 形式을 다루는 機構로서 byte register, word register, 및 long-word register 等이 必要하다. Z 8000에는 또 4倍長 register도 具備하고 있어 long-word 演算에 使用할 수 있다.

이 밖의 다른 데이터 形式들은 앞에 列擧한 데이터 形式 中의 어느 것으로 뒷받침 된다. 例를 들면, address는 long-word로서 다루고, 各 要素(segment number또는 offset) byte나 word로 다루어진다. 命令文은 1～5 word string으로 나타내고, program status는 4 words로 나타내는 것 等이다.

Z 8000 family가 앞으로 繼續 發展됨에 따라 새로운 데이터 形式에 對한 取扱도 可能해 질 것이다. Z 8000 構造는 앞으로 새로운 命令 set이 追加되더라도 이들을 充分히 다룰 수 있게 될 것이다.

iv) 命令(Instructions)

命令形式(instruction format)을 設計하는데 있어서는 制限된 bit들을 어떻게 opcode field, address mode field, 와 그 밖의 operand subfild 等에 割當하는가 하는 問題를 決定하여야 할 것이다. 이와 같은 instruction set format의 決定에 影響을 미치는 것은 무엇 보다도 命令使用總計(instruction usage statistics)이다.

命令形式;

Z 8000은 110個가 넘는 命令形式을 가지고 있다. 그 中에서 몇 가지 形式을 그림 2에 나타냈다. 이 그림에서 opcode field는 ADD, LD 等과 같은 命令의 形式을 指定한다. Mode field는 register R, direct address (DA) 等과 같은 addressing mode를 나타낸다. 여기에 data element type(W/B)와 register 指定 field를 添加하므로써 基本 命令 field

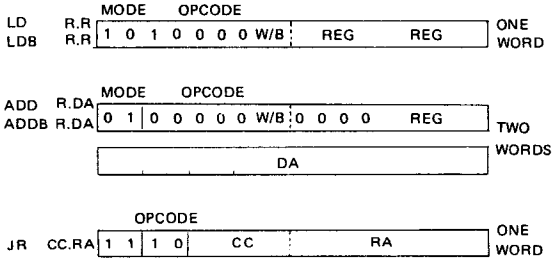

그림 2. 命令形式의 例(nonsegmented version)

가 完成된다. Long word로 된 命令은 byte形 命令 word形 命令의 경우와 다른 opcode를 使用한다. 頻繁히 쓰이는 命令들은 單一 word로 符號化되어 있고, 2個 以上의 演算子(operand)를 必要로하는 頻繁히 使用되지 않은 命令은 2個의 word로 構成되었다. Immediate value나 condition code(CC) 等과 같은 特別한 要素들에 對해서는 附加的인 field를 使用한다. 命令文 中에는 operand가 1個, 2個, 및 3個짜리가 모두 쓰인다. 4個의 operand를 갖는 것은

TRANSLATE AND TEST 命令 하나 뿐인데 이것은 또한 含蓄 레지스터 演算子(implied register operand)를 갖는 唯一한 경우이다.

Addressing modes:

Z8000은 8가지 addressing mode를 가지고 있다. 卽, register(R), indirect register(IR), direct address(DA), indexed(X), immediate(IM), base address(BA), base indexed(BX), 및 relative address(RA) 等이 可能하다. 이 밖에도 自動增加(autoincrement) 및 自動減少(auto-decrement)를 行하는 命令에서는 다른 addressing mode를 使用한다. 그러나 이것들 中에서 가장 많이 使用되는 것은 R, IR, DA, X, IM 等의 基本的인 5個 addressing mode이다. 2演算子型命令(two-operand instruction)의 경우 行先은 대개 register로 되어 있다.

Code 密度;

지금까지의 마이크로프로세서들은 初期型 pipeline構造로 制限되어 있기 때문에 處理速度는 實行되는 命令文의 數에 크게 依存하고 있다. 그러므로 code 密度는 프로그램의 크기를 減小시키기 위해서 뿐만 아니라 處理速度의 向上을 위해서도 重要한 問題로 되고 있다. 프로세서 設計者들은 가장 頻繁히 使用되는 命令들을 最少의 bit 數로 符號化하려고 하는 것이 普通이다. 그리고 基本的인 命令文의 길이는 word 單位로 增加하게 되어 있다. 그러므로 가장 頻繁하게 利用되는 1-operand形 및 2-operand形 命令文들은 單一 word 길이를 갖는다. 자주 使用되지 않은 命令文이나 2 operand 以上을 必要로 하는 命令文들은 적어도 2個의 word로 構成된다.

高水準信號處理能力;

마이크로프로세서 利用者에게는 어셈블리語를 使用할 때보다 高水準 言語를 使用할 수 있게 되면 컴퓨터 構造에 구애받지 않고 프로그램을 쉽게 作成할 수 있는 것은 當然한 事實이다. 特定 高水準 言語를 能率的으로 實行할 수 있게 하기 위해서는 컴퓨터 構造 역시 이에 適合하도록 되어 있어야 한다. 그러기 때문에 어떤 한 高水準 言語에 對해서는 매우 能率的인 프로세서 構造가 다른 言語에 對해서는 非能率的인 것이 될 수도 있다. Z8000은 汎用 마이크로프로세서이기 때문에 compile 問題와 code 發生 問題를 가장 잘 解決해 주는 方向으로 設計되어 있다. 例를 들면 Z8000 addressing mode와 데이터形式에 있어서의 規則性等을 들 수 있다. 分割(segmentation) 方式에 依한 addressing 構造는 structured programming으로 부터 發生되는 處理를 뒷받침할 수 있어야 된다. Procedure stack 上의 파라미터와 變數들의 呼出은 base address와 base indexed address mode를 使用하던가 short offset address mode의 指標에 依하여 實行된다. 그리고 address arithmetic 은 INCREMENT BY 1 TO 16 및 DECREMENT BY 1 TO 16 命令에 依하여 뒷받침 되고 있다.

데이터의 試驗, 論理的 評價, 初期値賦與 및 데이터比較 높은 각각 TEST, TEST CONDITION CODED, LOAD IMMEDIATE INTO MEMORY, 및 COMPARE IMMEDIATE WITH MEMORY 命令에 依하여 實行될 수 있다. 컴파일러와 어셈블러들은 文字例들을 자주 取扱하게 되므로 Z8000에서는 이들을 위하여 TRANSLATE, TRANSLATE AND TEST, BLOCK COMPARE, 및 COMPARE STRING 命令들이 있으며 이들을 使用하면 言語處理速度를 相當히 빠르게 할 수 있다. 그리고 PUSH와 POP命令을 어떤 register라도 stack pointer로 使用할 수 있게 한다.

v) 分割(segmentation)

Code 發生과 data 呼出을 容易하게 하기 위하여 address는 다루기 쉽게 되어야 한다. 메모리를 直接呼出(direct access)하는 構造에서는 address arithmetic을 全體 address에 對하여 適用할 수 있게 하기 위하여 linear address space를 使用한다. 이런 경우에는 address들은 같은 크기의 data type의 하나로 다루게 된다. 이렇게 하므로써 address를 새로운 data type으로 區別할 必要가 없게 된다. 이와는 反對로, Z8000은 非線型 address space를 가지고 있다. 즉, address는 7-bit segment 番號와 16-bit offset 番號의 2個 部分으로 되어 있다. 그리고 offset 番號만을 address arithmetic에 使用한다. Segment番號는 事實上 全 address space 中 한 部分에 對한 pointer 역할을 하며, 0부터 64K byte 範圍에서 變한다. 分割된 address의 hardware的 表示는 long word 卽, register雙(그림 3)의 形態이며 이렇게 하므로써 address의 各 部分을 다루기 쉽게 되어 있다. 分割 address 方式은 大容量 메모리와 小容

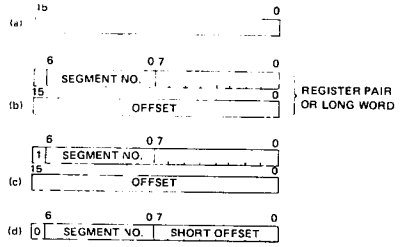

그림 3. Segmented address

量 메모리를 모두 能率的으로 뒷받침할 수 있는 方式中의 하나이다. Z8000의 두 가지 version ; 卽, 40-pin unsegmented version과 48-pin unsegmented version은 모두 構造的으로 並立性을 가지고 있고 이들 두 가지 應用間에 있어서의 擴張도 容易하게 되어 있다. 40-pin version의 各各의 64K byte address space는 48pin version의 segment 中에 하나가 될 수 있다. 각각의 40 pin version의 16-bit address는 그 segment 內에 있어서 offset가 되고 48-pin package version에는 40-pin version code가 實行될 수 있는 mode가 存在한다. Z80과 같은 現存 마이크로프로세서와의 並存性이 있고 Z8과 같은 새로운 마이크로컴퓨터도 Z8000과는 共通 segment 內의 外部데이터를 address할 수 있다.

Z8000의 hardware 性能도 또한 address segmentation에 依하여 改善되었다. Segment 番號는 address arithmetic에 包含되지 않으므로 address 計算의 結果가 나타나기 前에 bus로 내 보낼 수 있다. 이러한 特徵이 있기 때문에 MMU를 使用해도 memory 呼出時間에 지장을 주지 않고 CPU와 並行해서 動作할 수 있다. 그리고 indexing operation에 있어서도 16-bit 加算만 하기 때문에 過去보다 더욱 빨라졌다. Segment 番號와 offset를 區分하여 놓았기 때문에 software의 制限없이 短縮된 address를 使用할 수 있다. 短縮된 address에서는 256 byte 以下의 短縮된 offset를 使用할 수 있기 때문에 프로그램의 크기를 줄일 수 있게 된다. 끝으로 address space의 128 segment의 각각에 對한 메모리 保護와 大型시스템에서 要求되는 再配置 能力도 가지고 있다. 再配置(relocation)能力을 가지고 있으면 實際의 physical address에 구애받지 않고 logical address (그림 4)를 使用하여 應用프로그램을 作成할 수 있게 된다. 例를 들면, disk-based general data processing system을 多數人이 使用하고자 할 때에는 再配置 能力을 꼭 가지고 있어야 한다. 그러나 ROM에 기억된 命令만으로 動作하는 dedicated application에 있어서는 再配置는 必要하지 않다. 그리고 所要되는 全體 메모리 容量이 적을 때에도 再配置는 必要하지 않다.

結論的으로 分割된 address方式을 採擇하므로써 적은 費用으로 再配置, 프로그램保護, virtual memory 等이 쉽게 實現되었다. 이와 같은 問題들을 linear address space를 가지고 解決하려고 한다면 그 價格이 相當히 上昇함을 免치 못 할 것이다.

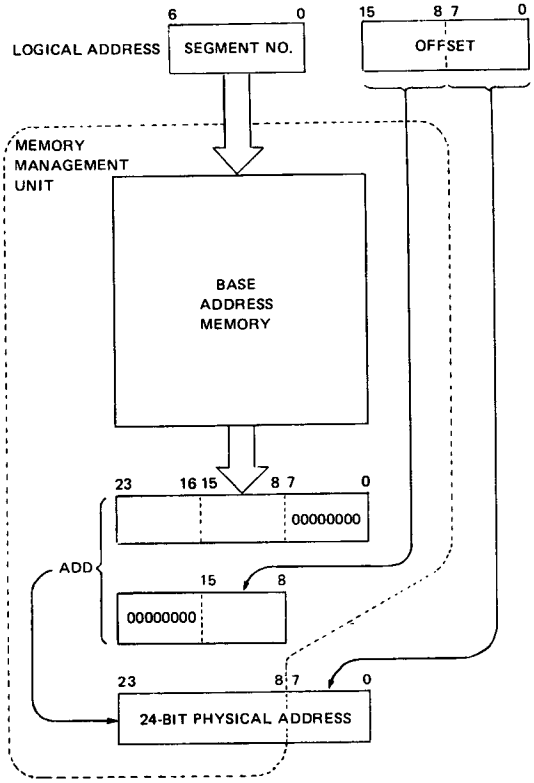

그림 4. Logical to physical address translation

參 考 文 獻

1. B.L. Peuto, "Architecture of a New Microprocessor," Computer Jour., IEEE. February 1979.
2. Zilog, Z8000 Technical Manual, Zilog Inc., 1979.
3. Zilog, MMU Technical Manual, Zilog Inc., 1979.