지식공학을 위한 하드웨어

黃 時 永
(正 會 員)
三星綜合技術院 情報시스템研究所 責任研究員

대다수의 사람들이 갖고 있지 않은 특정분야-예를 들어 의학, 법률, 과학적 분석, 회로설계 등-의 지식을 요하는 일은 이에 필요한 지식을 갖춘 전문가에 의해서만 가능하였으나 최근 컴퓨터 기술의 발달로 이런 일을 해주는 프로그램의 탄생을 보게 되었으며 이런 프로그램을 전문가 시스템이라고 하고 전문가 시스템을 구축하는 일을 지식공학(knowledge engineering)이라고 한다. 지식공학은 인공지능을 응용면에서 추구하는 바이며 지적기능을 프로그램화하는 과학이라 할 수 있다. 먼저 지식공학을 위한 컴퓨터 구조를 기술하기 전에 인공지능 응용 프로그램의 특징에 대해 언급하고자 한다.

Ⅰ. 인공지능 응용 프로그램의 특징

대부분의 AI 알고리즘은 비확정성(nondeterminism)을 가지므로 어떤 절차(procedure)가 수행되어야 하는지 알 수가 없으며 이는 지식의 결핍과 문제의 불완전한 이해에서 오는 것으로 문제를 해결하기 위해서는 모든 경우의 수를 고려해야만 한다. 이런 불완전한 지식과 해결과정의 불확실성 때문에 이미 기억장치에 있는 자료구조 및 함수가 사용될 것인지, 아니면 새로운 자료구조와 함수가 생성될지 알 수가 없다. 더구나 사용될 자료구조의 최대 크기를 알 수가 없으므로 미리 기억장치를 할당하는 것이 불가능하다. 따라서 대부분의 AI 프로그램은 동적수행(dynamic execution)을 필요로 하고 있다. 뿐만 아니라 AI 프로그램은 고도의 비확정성으로 인해 병렬처리와 분산처리가 강력히 요구되고 있다.

지식은 사실(fact), 가정과 믿음(assumption and belief) 그리고 heuristic 규칙으로 나눌 수 있으며 보다 쓸모있는 지식은 문제해결을 위한 탐색시간을 줄여줌으로써 주어진 문제를 쉽게 풀 수 있게 해 주는 역할을 한다. 대부분의 AI 문제를 보면 상당한 복잡성이 내재되어 있기에 유용한 정보가 너무 많을 뿐 아니라 얻은 지식이 애매모호하거나 불확실한 경우도 있다. 따라서 지식의 관리와 표현이 매우 중요한 요소라 할 수 있다. 마지막으로, 환경이 변할 가능성도 있으며 이는 설계시에는 예상할 수 없으므로 대부분의 AI 프로그램에서는 문제를 해결하기 위한 지식이 불완전할 경우가 많이 발생한다. 그러므로 AI 시스템은 계속적인 갱신과 새로운 지식의 습득을 허용할 수 있게 설계되어야 한다.

지식의 습득, 표현 그리고 지적인 사용은 AI 시스템에서 기본적인 것이라 할 수 있으며, 지능을 갖춘 시스템은 화상, 음성, 지면 자료로부터 필요한 정보를 추출해 낼 수 있어야 한다. 또한 이들 자료는 불완전하거나 부정확하거나 혹은 서로 모순될 수 있으므로 지식처리 기능에는 화상, 음성, 지면자료에 대한 적절한 인식과 이해가 필요하다. 뿐만 아니라 지식처리는 지식자체를 표현하는 점, 탐색(search) 기능에 의존적이라는 점, 국부적인 처리에 국한되지 않는 대용량의 기억장치를 요구한다는 점, 다양한 크기의 메시지를 처리하는 점, 비확정성을 사용한다는 점, 대화형식의 입·출력을 한다는 점 및 지식 데이타 베이스를 사용한다는 점에서 종래의 수치계산 처리방식과는 판이하게 다른 특성을 갖고 있다. 이런 특성때문에 수치계산용인 종래의 컴퓨터의 구조적 특성으로는 효율적인 지식처리를 하기에 부적당하게 되었으며 결국 새로운 컴퓨터 구조를 추구하게 되었고, 특히 과거 10년간의 VLSI 기술의 발달은 새로

운 인공지능 컴퓨터를 가능하게 해주었다. 그림 1 에서 보는 바와 같이, AI 컴퓨터의 설계는 bottom-up 방식과 top-down 방식의 두가지로 대별되는데 전자는 아키텍춰에서 시작하여 이 구조적 특징이 제공하는 응용쪽으로 나가는 것이고, 후자는 그 반대라 할 수 있다. 전자의 예로서 자료검색, 패턴매칭, 그리고 unification을 하는 하드웨어를 기반으로하여 resolution에 기초를 둔 추론 시스템을 구축할 수 있을 것이다. 여기서 주의해야 할 점은 두 방법 모두 인공지능언어(AI programming language), 병렬 알고리즘, 분산운영체제 및 분산처리기법, 지식획득 기능 및 추론엔진 그리고 지적 맨-머신 인터페이스(man-machine interface)에 중점을 두고 있는 점이다. 그림1의 각 단계에서의 석택은 결코 독립적이거나 배타적일 수 없다. 예를 들어 어떤 전문가 시스템은 rule에 기초를 둔 추론과 resolution에 기초를 둔 추론을 동시에 사용하도록 구현될 수 있다는 것이다.

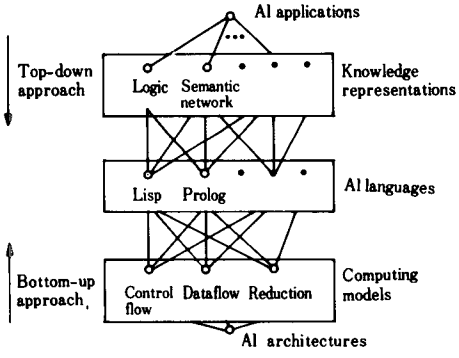


그림 1. AI 컴퓨터의 설계방식

다음으로 AI 머신을 위한 하드웨어를 마이크로 수준(micro-level), 마크로 수준(macro-level), 그리고 시스템 수준에서 살펴보기로 한다. 마이크로 수준에서는 기본적인 오퍼레이션(primitive operation)을 하드웨어화한 것으로 unification 하드웨어, 동적 데이타 형태 체크(dynamic data type checking)를 위한 tag bit, 그리고 하드웨어 스택 등이 있으며 그 위 단계인 마크로 수준은 마이크로 수준의 primitive로 구성되어 좀 더 복잡한 기능을 하기 위한 것으로서 좀 더 복잡한 기능을 하기 위한 것으로서 데이타 베이스 머신, searching을 위한 하드웨어와 Garbage collection 하드웨어 등이 있다. 또한 시스템 수준은 AI 언어를 보조하기 위한 하드웨어로 구성된다. 결국 지식공학을 위한 하드웨어를 AI 컴퓨터라고 정의할때 이 AI 컴퓨터는 크게 언어에 기초를 둔 시스템, 지식에 기초를 둔 시스템 그리고 맨-머신 인터페이스에 기초를 둔 시스템으로 대별될 수 있으며 여기서는 언어에 기초를 둔 시스템에 대해, 특히 LISP 언어와 PROLOG 언어머신에 대해 살펴보기로 한다. Ⅱ장에서는 LISP머신인 심볼릭스(symbolics), Ⅲ장에서는 prolog머신, Ⅳ장에서는 지식공학을 위한 시스템인 일본의 제5세대 컴퓨터에 대해 언급하고자 한다.

Ⅱ. LISP 머신

LISP 프로그램은 함수(function)들의 집합으로 볼 수 있으며 이들 함수들을 동시에 수행함으로써 고도의 병렬성(parallelism)을 갖게 된다. 뿐만아니라 LISP의 응용성(applicativeness) 및 반복성(recursiveness)에 의하여 효율적인 스택이 필요하게 되며, 기본 자료구조로 linked list를 사용함으로써 Garbage collection의 효율을 높여주는 자동기억장치할당 (automatic storage allocation) 메카니즘을 제공해 준다. 그러나 이들 기능들이 종래의 컴퓨터에서 수행되기에는 적합치 않은 점이 많으며 특히 자료타입 (data type)을 선언하지 않는 LISP의 자료객체(data object)를 효율적으로 수행시키는 것은 불가능하다고 할 수 있다. 따라서 LISP의 빠른 수행을 위한 새로운 컴퓨터구조가 필요하게 되었다.

심볼릭스도 다른 컴퓨터 시스템과 마찬가지로 여러 단계로, 혹은 여러 형태의 구조(architecture)로 구성된다. 여기서는 심볼릭스 컴퓨터에서 정의된 3개의 구조 즉 시스템 구조, 인스트럭션 구조, 그리고 프로세서 구조에 대해 알아 보고자 한다.

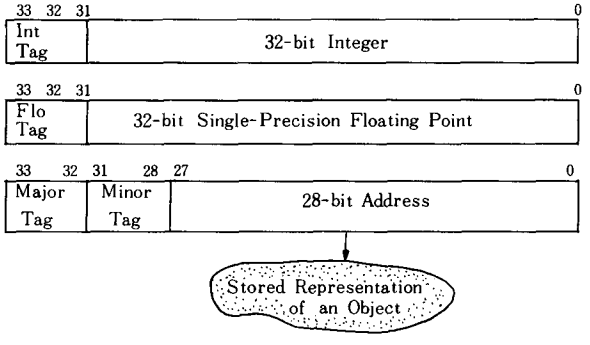
시스템 구조란 시스템이 사용자나 프로그래머에게 어떻게 보이는지를 정의하게 되며 주로 소프트웨어로 구현된다. (여기서 하드웨어는 단지 무엇이 가능한지 또 무엇이 불가능한지에 대한 경계라고 말할 수 있다.) 심볼릭스 시스템 구조는 소프트웨어의 생산성을 높이고 AI 분야의(AI 이외의 분야도 포함) 커다란 프로그램을 효율적으로 수행시켜 주며 특히 이제까지는 프로그램하기 어려웠던 응용 분야에 적합한 고차원의 사용자 인터페이스 및 객체지향형(object-oriented) 프로그래밍 환경을 제공할 뿐 아니라 배열의 첨자한계, 함수에 전달되는 매개변수 및 데이타 형태(data type)의 수행시간 체크를 해준다. 또

한 임의의 프로그램이 수행되고 있는 중에도 그 프로그램이 점진적으로 변경될 수 있으며 debugging시에 필요한 정보를 잃지 않고 보존하는 기능이 있다. 특히 기억장소 관리(memory management)가 매우 효율적이며 프로그래머와 무관하게 자율적으로 처리된다.

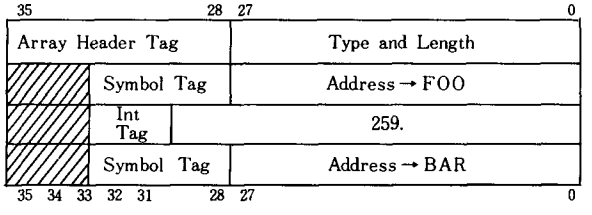
심볼릭스는 안전성을 높이기 위하여 처리속도를 늦게하지 않으며, 이는 계산 및 기억장치 접근과 동시에 하드웨어에 의한 low-level 체킹이 수행되며, 오퍼랜드의 형태에 관계없는 혼합연산이 가능하고, 함수(function) 호출이 매우 빠를 뿐 아니라 응용 프로그램에서 가상기억장치 페이징(paging)을 지정할 수 있기 때문이다. 이러한 기능들에 의하여 심볼릭스 시스템은 프로그래머가 사소한 점에 신경쓰지 않고서도 프로그램을 할 수 있는 환경을 제공하고 있다.

다음으로 인스트럭션 구조에 대해 알아본다. 심볼릭스는 서로 다른 구조의 독립적 변경이 가능하도록 설계되었다. 사용자는 단순히 시스템 구조만 보게되고 인스트럭션 구조와 같은 하위 단계는 사용가능한 최신기술에 의하여 성능을 극대화시키고 가격을 극소화시킬 수 있도록 자유로운 변경이 가능하게 되어 있다.

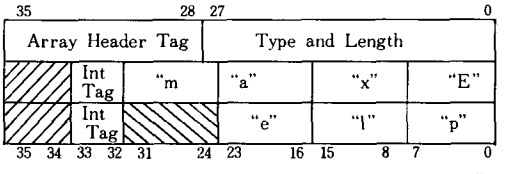
LISP 시스템에서 사용되는 자료의 기본 형태는 개념적 객체(conceptual object)를 지정하는 object reference이며 이 object reference는 32 bit의 데이타 워드와 데이타의 해석 방법을 결정하는 2 bit의 데이타 타입 태그로 구성된다. 그림 2에서 보는 바와 같이 태그는 데이타 워드의 형태에 따라 가변적 길이를 가지며 이 가변 길이의 태깅방법은 32 bit 의 고정소수점 수와 부동소수점 수를 나타내는데 아무런 지장도 주지 않는다. 또한 이 태깅 메카니즘을 사용함으로써 모든 자료의 자체설명(self-description)이 가능하며, 자료의 형태, 배열의 첨자한계, 그리고 정의되지 않는 함수와 변수의 수행시 이를 체크



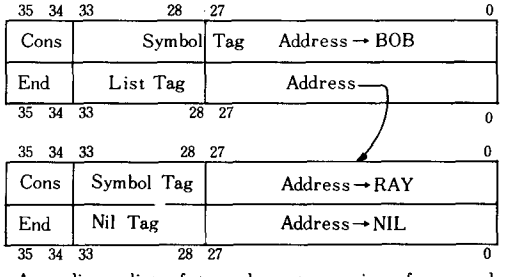
An object reference is a 34-bit quantity, consisting either of a 32-bit data word with a 2-bit data type tag, or of a 28-bit address with a 6-bit data type tag.



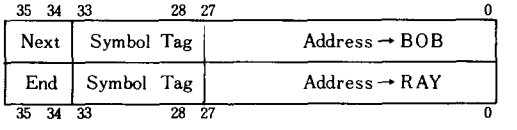
An array of three elements—FOO, 259, and BAR—consists of a header word defining the type and length of the array, followed by an object reference for each array element.



A string containing the seven characters "Example" stores each character in a single 8-bit bxyte. Bytes are packed into 32- bit integer objects.



An ordinary list of two elements requires four words of storage. Unlike arrays, lists do not have headers.



A compact list of two elements requires two words of storage. It uses the cdr code to eliminate two object references.

그림 2. 심볼릭스 컴퓨터의 자료구조

하기 위한 정보가 항상 유용한 상태로 보존될 수 있고, 워드부분과 태그부분의 병렬수행이 가능함으로써 안정성과 빠른 처리속도를 동시에 만족시킬 수 있다. 뿐만아니라 혼합연산 명령어는 오퍼랜드의 태그에 따라 오퍼레이션을 바꿀 수 있으며 자동기억장치할당이 간단해지고 효율화 되므로써 하드웨어로 처리되는 것을 가능하게 해준다.

심볼릭스에서 하드웨어 및 firmware에 의해 수행되는 명령어들은 17 bit으로 구성되어 있으며 9 bit는 오퍼레이션을 지정하고 나머지 8 bit는 오퍼랜드를 나타낸다. 여기서 두 명령어가 한 워드를 구성함으로써 프로그램 크기를 줄일 수 있으며 프로세서 속도에 대한 명령어 fetch 밴드위드의 비율을 적게하므로써 기억장치를 효율적으로 사용할 수 있다.

명령어들은 스택을 이용하여 수행되며 오퍼랜드를 스택에서 꺼낸후 처리결과를 스택에 다시 넣는 0-주소 연산 명령어와 스택의 어떤 위치든지 주소로 접근할 수 있게 해주는 1-주소 연산 명령어가 있다. 많은 명령어가 하드웨어나 firmware에 의해 직접 구현된 LISP 함수로서 내장함수(built-ins)라고 불리며 이 내장함수에는 eq, symbolics, logand, car, cons, member 그리고 arcf 등이 있다. 여기서 어떤 기능을 하드웨어화 하느냐는 하드웨어화에 의한 성능향상의 정도에 달려있으며 성능향상에 도움이 안되는 것은 소프트웨어로 남겨두어 변경이나 debugging이 쉽게 되도록 하고 있다.

심볼릭스의 명령어는 error나 exception에 대한 점검이 시스템에 의하여 자동처리되므로 프로그램에서 이를 점검하는 별도의 명령어를 사용할 필요가 없다. 다른 머신과는 달리 심볼릭스는 indexed나 indirect addressing 방법을 사용하지 않으며 그 대신에 배열을 첨자를 통해 접근하고 list의 car부분을 가져오는 구조화된 객체지향 연산(object-oriented operation)을 하는 명령어들을 가지고 있다.

LISP 프로그램은 전술한 바와 같이 함수의 집합이다. 따라서 함수의 호출과 복귀를 가능한 한 빠르게 하는 것이 매우 중요하다. 심볼릭스의 함수 호출 메카니즘은 기억장치 조회 회수를 줄이기 위해 스택버퍼를 사용하고 있으며 예외 처리를 빨리하고 수행시간에 대한 의사결정을 간단히 하기 위한 벡터(vector)를 둠으로써 더 빠른 속도를 얻는다.

마지막으로 프로세서 구조에 대해 알아본다. 심볼릭스에서는 성능향상을 위하여 명령어 수행에 드는 클럭 싸이클 수를 줄임으로써 에러(error)와 예외사항(exception)에 대한 점검이 병렬로 수행되도록 설계되어 있다. 명령어들은 그림 3에서 보는 바와같이 수평마이크로코드(horizontal microcode)의 제어하에서 4 단계의 pipeline에 의해 처리된다. 데이타 패스는 병렬로 동작되는 여러 unit을 연결해주며 데이타의 이동, 산술과 논리 연산 명령어등은 한 클럭 싸이클에 수행된다. 예를 들면 Add명령어 수행시 다음과 같은 작동이 병렬로 수행된다. 스택버퍼에 두 오퍼랜드를 옮기고 고정소수점 산술기는 32 bit의 덧셈 연산을 하여 오버플로우가 일어났는지 점검하며, 태그 프로세서가 오퍼랜드의 데이타 타입을 점검하고 스택버퍼는 산술기로부터 결과를 얻으며, 다음 명령어를 디코드하여 이 명령어의 수행을 제어하기 위한 마이크로 명령어를 생산한다.

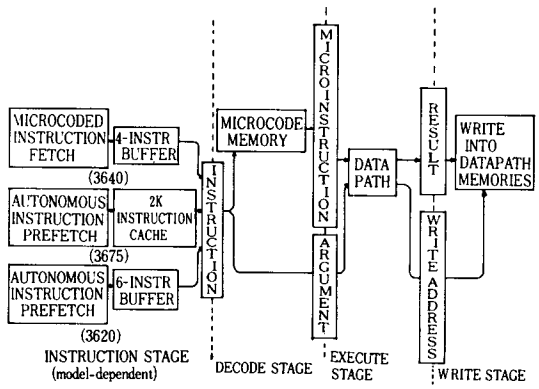


그림 3. 심볼릭스 컴퓨터의 pipeline 구조

이상에서 심볼릭스의 특징을 통해 AI 컴퓨터 즉 지식공학을 위한 하드웨어 지원 시스템이 어떤 기능을 갖고 있어야 하는지 간략히 다루어 보았다. 다음 장에서는 실험단계에 있는 prolog 머신의 구조를 살펴보기로 한다.

Ⅲ. PROLOG 머신

이 장에서는 RPM(reduced instruction set prolog machine)에 대해 알아본다. 프롤로그 언어는 여러 응용분야에 맞는 강력한 프로그래밍 언어로 인정받고 있으며 전문가 시스템은 물론 제5세대 컴퓨터의 주요 언어로 제안된 바 있다. 따라서 이처럼 강력한 프롤로그 프로그램을 효율적으로 수행시키기 위한

머신의 설계가 요구되고 있다.

프롤로그 프로그램은 다음의 세가지 다른 오퍼레이션에 의하여 수행된다. 즉 clause의 인덱싱(indexing)이나 sub-goal 사이의 순차화(sequencing) 등의 clause 제어 오퍼레이션, unify 오퍼레이션, 그리고 산술·논리 연산의 수행을 위한 함수 오퍼레이션에 의하여 수행된다. 이들 오퍼레이션들은 각각의 수행을 전담하는 unit을 둠으로써 각 unit을 간단하고 효율적으로 설계할 수 있을 뿐 아니라 동시수행 (concurrent operation)으로 성능향상을 얻을 수 있으며, AND/OR 병렬성이나 unify 오퍼레이션을 위한 unit을 첨가함으로써 시스템의 확장성을 높일 수도 있다.

그림 4 의 RPM 머신은 컴파일된 프롤로그 프로그램을 수행시키기 위한 co-processor 구조를 갖고 있으며 이는 동시에 수행되는 여러 unit으로 구성되어 있다. 각 unit들의 역할은 다음과 같다.

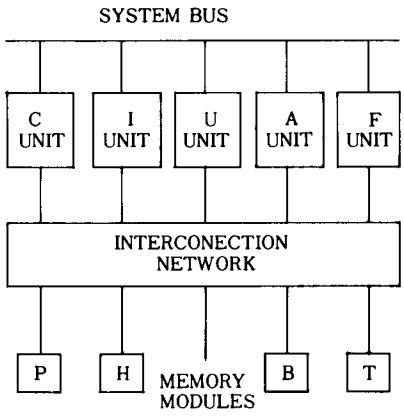


그림 4. RPM 구조

Control unit(c-unit)은 명령어를 수행시키고 다른 unit의 오퍼레이션을 제어하며 clause indexing, goal간의 sequencing 및 F-unit이나 U-unit을 활성화(activation)시키는 역할을 한다. Unification unit (u-unit)은 unification 오퍼레이션을 신속히 수행시켜주는 하드웨어 unit으로서 I-unit 및 A-unit 두개의 prefetching unit을 갖고 있다. 명령어 fetch unit (I-unit)은 명령어를 기억장치로부터 fetch하여 C-unit이 이전의 명령어를 수행하는 동안 이를 디코드하며, A-unit은 주소와 자료를 U-unit에 전달한다. 이외에도 산술·논리 연산을 하는 functional unit (F-unit) 및 상호연결 네트워크(interconnection network)를 통해 동시에 접근할 수 있는 8개의 기억장치 모듈이 있다. RPM에는 위의 unit들 외에도 시스템 버스 제어 unit과 I/O unit이 있으며 각 unit들은 시스템 버스를 통해 데이타를 교환한다.

프롤로그 프로그램이 RPM의 내부 목적코드(object-code)로 컴파일 되었을때 이 목적코드는 주기억장치내의 프로그램 영역(program area)에 존재하며 프롤로그 프로그램 수행시에 쓰이는 자료와 상태정보등은 서로 다른 기억장치 세그먼트에 존재하게 된다. 여기서 잠시 기억장치 구조에 대해 언급하기로 한다. 그림 4에서 P는 프로그램 장소로서 이곳에는 컴파일된 프롤로그 프로그램이 입력되게 되고 H에는 clause의 주소와 데이타가 저장되며 B와 T는 스택으로서 B에는 backtracking시 다시 환원되어야 하는 상태정보를 담고 있는 backtracking 환경 (environment)이 저장되며 T에는 계속 수행되어야 하는 goal의 상태정보가 저장된다.

RPM의 인스트럭션 집합에는 cut(backtracking을 방지하기 위한 primitive)및 연산 오퍼레이션을 구현하기 위한 메카니즘이 제공된다. 이는 간단한 컴파일 프로세스를 제공하며 여러 RPM의 functional unit들이 동시에 수행될 수 있도록 해준다.

RPM의 인스트럭션은 오퍼레이션 코드와 하나 또는 여러개의 오퍼랜드로 구성되며 goal들간의 순서와 clause 환경의 운영을 맡는 절차 명령어 (procedural instruction), clause의 효율적 선택을 위한 인덱싱 명령어(indexing instruction), 수치·논리연산을 위한 연산 명령어(arithmetic instruction), unify 오퍼레이션을 행하는 unify 명령어 그리고 입출력 장치를 제어하기 위한 기타 명령어등으로 나누어진다.

Ⅱ장과 Ⅲ장에서는 AI 언어인 LISP와 prolog를 지원하는 컴퓨터 시스템의 특징에 대해 살펴보았다. Ⅳ장에서는 실제적인 지식공학을 지원하는 하드웨어라 할 수 있는 일본의 제5세대 컴퓨터 시스템 구조에 대해 언급하고자 한다.

Ⅳ. 제5세대 컴퓨터 시스템

컴퓨터 기술은 탄생시점부터 계속적으로 고속도의 연산처리 능력과 대규모 기억용량을 추구해 왔으며 주로 수치계산을 위해 발전되어 왔다. 결과적으로 컴퓨터는 입력과 출력에 커다란 제약을 갖게 됨으로써 그 응용에도 제약을 주게 되었다. 한편 컴퓨터의 응

용 소프트웨어가 확산됨에따라 인간의 언어, 이미지(image), 그림 등과 같은 자연스러운 정보전달방법으로 입·출력을 하고자 하는 커다란 요구가 생기게 되었으며 이러한 요구는 컴퓨터 지능과 같은 새로운 기능과 이를 위한 컴퓨터 구조에 관한 새로운 기술적 기초를 만들게 되었다. 여기에는 VLSI 기술, 대용량의 기억장치 실현기술, 고속처리 능력개발에 대한 가능성, 인공지능과 패턴인식의 가능성, 그리고 통신과 정보처리의 기술적 융합등이 포함된다.

그러면 지능을 가진 컴퓨터에서는 어떤 기능이 요구되고 있는가? 우선 말, 그래픽, 이미지와 문서를 통해 정보를 입력하고 출력할 수 있도록 하는 기능과 일상용어로 정보처리를 가능하게 하는 기능이 필요하다. 또 여러 분야의 문제해결을 위한 도구로써 컴퓨터를 보다 효율적으로 사용하기 위해서는 특정 분야의 지식이 저장되어 실제로 사용되어지는 기능이 있어야 하며 이를 위하여 사람에 의해 주어지는 애매모호한 요구에 대응할 수 있도록 지식을 충분히 습득하고 활용할 수 있는 학습 및 추론기능이 필요하다.

그러나 이러한 시스템의 실현은 처리속도와 기억장치의 성능향상 없이는 이루어질 수 없는 것이다. 즉 VLSI 기술에 매우 의존적이어서 silicon 소자보다는 Josephson 소자나 GaAs 소자와 같은 더 빠른 소자의 사용이 요구되며 이런 소자를 이용하여 고속의 처리능력이 생기게 되면 컴퓨터의 성능향상이 이루어지게 된다는 것은 분명한 일이다. 더우기 Josephson 소자는 정보를 저장하는데 거의 에너지가 필요하지 않다는 장점도 있으며 또한 광학기술은 입출력장치, 자료전송 그리고 지식베이스 저장도구와 같은 보조기억장치등 그 응용분야가 넓다. 하지만 단순히 VLSI에 의해 얻을 수 있는 속도는 한정되어 있으므로 pipeline 과 SIMD 등을 이용한 병렬처리기법 (parallel processing) 이 발전하고 있으며 associative memory와 같은 논리와 기억장치가 결합된 기억장치 시스템도 발달하고 있다.

지식공학을 위한 제 5 세대 컴퓨터는 전술한 기능을 갖기 위해 추론머신, 지식베이스머신, 그리고 지적 인터페이스머신으로 구성되며 이는 종래 컴퓨터의 중앙처리장치, 가상기억장치와 화일 시스템을 갖는 주기억장치, 입·출력 채널과 입출력장치에 각각 해당된다. 그림 5 는 제 5 세대 컴퓨터의 구조를 표시하고 있다.

추론머신은 마치 LISP 머신이나 prolog 머신과 같은 고수준의 언어머신으로 생각할 수 있다. 이는 VLSI 기술에 의한 고도의 병렬처리가 가능한 데이타플로우 머신으로서 객체지향형 프로그래밍 언어를 지원하고 semantic gap 을 줄이는 역할을 하게 된다. 지식베이스 머신은 추론머신을 위한 기억장치관리 시스템의 역할을 하는 관계 데이타베이스 머신이며 지적 인터페이스 머신은 일상어, 그래프등의 인간에게 자연스러운 의사표현방식으로 컴퓨터와 입·출력하게 해주는 머신이라 할 수 있다.

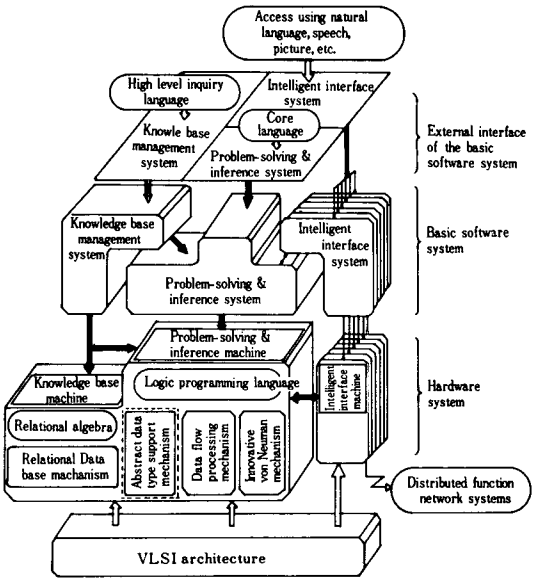


그림 5. 제 5 세대 컴퓨터의 기본구조

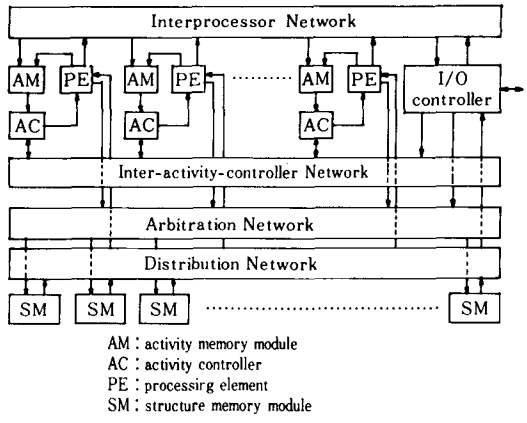
추론머신은 추론기능을 신속히 처리하기 위한 심볼릭 프로세싱 기능을 가지며 패턴매칭, 비확정성에 대한 처리기능, Garbage collection 기능등을 갖는다. 제 5 세대 컴퓨터에서는 추론머신의 언어로서 prolog를 사용하는데 그 이유는 prolog가 LISP의 확장으로 볼 수 있고 Lisp 외에 Smalltalk 이나 APL과 같은 언어의 좋은 점을 통합하였기 때문이다. Prolog는 논리에 바탕을 둔 프로그래밍 언어로 관계 데이타베이스와 같은 논리적 바탕을 갖고 있으며 프로그래밍과 데이타베이스 query 언어를 통합하는 기본 언어로 적당할뿐 아니라 자연어 처리와 고차원의 추론에 적당하다는 장점도 있다. 즉 prolog가 추론 오퍼레이션에 기초를 둔 언어이기에 prolog 머신이 추론머신에 한층 가깝다고 볼 수 있다.

그러면 과연 추론머신의 하드웨어는 어떤 것일까? 그것은 아마도 데이타플로우 머신이 될 것이다. 추론 시스템에서 비확정성의 문제는 피할 수 없는 것이고 현재의 순차처리방식의 컴퓨터(sequential computer)에서는 backtracking으로써 이를 해결하고 있으나 이 방법은 매우 비효율적일뿐 아니라 이러한 비확정성의 문제는 병렬 오퍼레이션의 범주에 드는 것이므로 데이타플로우 머신이 추론 시스템으로 가장 적합하다고 할 수 있다. 또한 데이타플로우 머신의 중요한 특징 중의 하나는 병렬 프로세싱 구조와 고수준의 프로그래밍 언어를 결합할 수 있게 되므로써 종래의 병렬 프로세서가 갖고 있던 semantic gap를 없앨 수 있다는 점이다. 일반적으로 추론 메카니즘에 많은 병렬성이 존재한다 할지라도 추론과정은 행렬연산의 계산과정보다 훨씬 불규칙적이며 이런 불규칙한 연산에 적용하여 강력한 추론기능을 수행하는데는 데이타플로우 머신이 적합하다고 할 수 있을 것이다. 비록 현재까지는 데이타플로우 머신과 논리 프로그래밍 모델사이에 차이가 존재하지만 가까운 시일내에 이는 사라질 것으로 보인다.

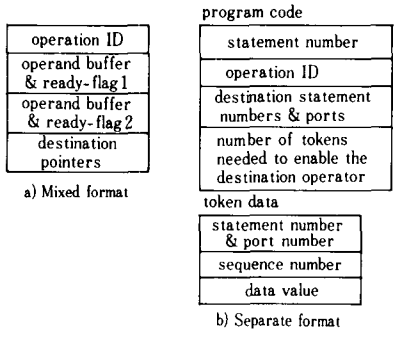
데이타베이스 머신에서는 관계대수 프로세서(relational algebraic processor)가 추론머신이므로 이 역시 데이타플로우 머신의 구조를 갖게 될 것이다. 데이타플로우 머신의 구조, 프로그램 및 자료의 예는 그림 6 과 같다.

지식베이스 머신은 고도의 편리한 맨-머신 인터페이스를 갖고 있으며 거대한 양의 상식과 전문지식을 갖고 있다. 지식베이스 머신의 효율성이 전체 시스템의 효율성에 지대한 영향을 미치므로 지식베이스 관리 메카니즘은 지식자료를 효율적으로 저장하고 검색하는 하드웨어 시스템으로 구현되어 진다. 지식베이스 관리 메카니즘은 지식베이스 머신에 의해 주어진 정보가 추론과정을 제어하는 추론머신과 직접적으로 인터페이스 되도록 하는 역할을 갖는다. 지식베이스 머신은 관계 데이타베이스 모델에 기초를 둔 데이타베이스 머신과 추론머신과 협력하여 데이타베이스 머신을 제어하는 메카니즘으로 구성되며 전자는 관계대수연산을 포함한 관계데이타베이스 머신이고 후자는 추론머신에 인터페이스하는 메카니즘이라 할 수 있다. 또 고도의 기능을 빠른 속도로 처리하기 위해 병렬처리와 분산처리가 기본개념이 되는데 이는 데이타플로우 머신과 일치한다.

지적 인터페이스 머신은 제5세대 컴퓨터의 기능적 이미지를 보여주는 것으로써 일상용어, 글, 그림



(a) A configuration of a data flow machine



(b) An example of the program and data format

그림 6. Demand-driven 방식의 dataflow 머신

등 인간에게 자연스러운 통신수단을 써서 컴퓨터와 대화함으로써 사용자가 컴퓨터를 사용하기 쉽게 해주는 기능을 갖는다.

V. 맺 음 말

이제까지 세 종류의 지식공학용 하드웨어 시스템에 대해 알아보았다. 대부분이 아직 실험단계에 있으며 간단하면서 자주 쓰이는 알고리즘은 하드웨어화하고 기억장치와 프로세싱 unit을 결합시켜 지적 기억장치를 만들어 나가는 추세이다. 또한 응용 프로그램의 효율적인 처리를 위해 보다 나은 알고리즘이 우선적으로 요구되며 보다 진보된 소프트웨어 관리기법과 보다 나은 하드웨어 아키텍춰가 필수적으로 요구되고 있다. 결국 AI 프로그램 대다수가 heuristics에 기초를 두고 있기 때문에 보다 나은 지식

표현에 기초를 둔 heuristics 정보를 사용함으로써 하드웨어에 의한 성능향상보다 더 뛰어난 성과를 기대할 수 있으며 아주 복잡한 문제는 heuristics와 병렬처리기법을 결합시킴으로써 해결해 나가야 할 것이다.

參 考 文 獻

[1] David A. Moon, "Symbolics architcuture," *IEEE Computher,* vol. 20, no. 1, pp. 43-52, Jan. 1987.

[2] Benjamin W. Wah, "New computers for AI processing," *IEEE Computer,* vol. 20, no. 1, pp. 10-15, Jan. 1987.

[3] Kai Hwang, Joydeep Ghosh, and Raymond Chowkwanyun, "Computer architectures for artificial intelligence processing," vol. 20. no. 1, pp. 19-26, Jan. 1987.

[4] J. Vlahavas and C. Halatsis, "A RISC prolog Machine Architecture," Microprocessing and Microprogramming, vol. 21, pp. 259-266, August 1987.

[5] T. Moto-Oka, Edlitor, "Fifth Generation Computer System," North-Holland.

[6] Proceedings of the International Conference on Fifth Generation Computer Systems 1984, November 1984.

♣ 行 事 案 內 ♣

본 학회 기술세미나 계획(7월 중 실시 예정)을 아래와 같이 알려드리오니 이에 관심 있으신 회원 여러분의 많은 참여 있으시기를 바랍니다.

행 사 명	일 시	장 소	비 고
"UNIX" 단기강좌	7월경	한국과학기술원	전자계산연구회 주 관
"통신용 보안기 및 Noise filter" 단기강좌	7월 12일	한국자동차공업회관	
"공통선 신호방식 (C. C. S)" 워크샵	7월 28일	한국전기통신공사	전자교환연구회 주 관
"VLSI Digital Signal Process" 단기강좌	7월 29일 ~30일	한국자동차공업회관	음향 및 신호처리 연구회 주 관

* 위에 대한 자세한 사항은 학회 사무국(568-7800, 7489)으로 문의하시기 바랍니다.