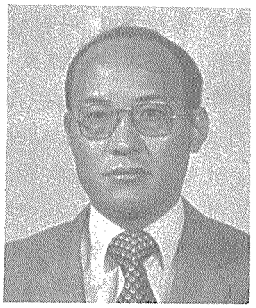
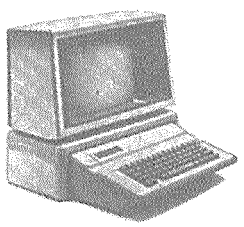


吳 吉 祿
韓國電子通信研究所
컴퓨터연구부장 / 工博

IBM/370의 에뮬레이션을 위한 마이크로 인스트럭션 포맷 설계



1. 서론

디지탈 컴퓨터의 CPU제어 방법으로 1951년에 M. V. Wilkes[1]에 의해 제안된 마이크로 프로그래밍 기법이 많이 이용되고 있다. 그 이유는, 예전의 hard-wired 제어 방법에 의한 CPU 제어방법보다 제어 장치의 설계를 체계화 할 수 있고, 컴퓨터 구조를 쉽게 변경시킬 수 있으며, 제어 기억장치를 WCS(Writable Control Storage)로 구현할 경우 새로운 기계어의 추가가 용이하며, 소프트웨어의 호환성, 경제성 및 고장 진단성의 장점을 주기 때문이다[2, 3]. 마이크로 프로그램에 의한 CPU제어방식이 이와 같은 여러 장점을 가지려면 CPU를 효율적으로 제어할 수 있는 마이크로 인스트럭션의 설계가 필요하다. 왜냐하면, 마이크로 프로그램에 의한 제어 방식을 가진 모든 디지탈 컴퓨터의 각 기계어들은 마이크로 인스트럭션들의 조합에 의해 fetch, 디코드되어 그것들이 의미하는 오퍼레이션을 수행하게 되기 때문이다. 따라서 마이크로 인스트럭션은 기본 동작으로 시스템의 성능에 큰 영향을 주므로 하드웨어 자원을 효율적으로 제어할 수 있게 설계되어야 한다[4, 5, 6, 7]. 그런데 CPU의 제어에 사용되는 마이크로 인스트럭션들은 마이크로 인스트럭션 포맷의 임의의 필드들의 조합으로 정의되어지므로 마이크로 인스트럭션 포맷의 설계가 디지탈 설계에서 컴퓨터의 중요한 부분을 차지한다.

본 논문에서는, 한국전자기술연구소에서 32비트 미니 컴퓨터 개발 프로젝트에서 설계된 CPU[8]를 제어하여 IBM/370 model 138을 기능적으로 동등하게 에뮬레이션하기 위한 마이크로 인스트럭션 포맷을 설계하였다. 그 과정에서 target machine(TM)인 IBM/370을 분석해서 TM을 에뮬레이션할 host machine과 자원 맵핑을 한 뒤, host machine을 제어하기 위한 마이

크로 인스트럭션 포맷을 설계하였다.

2. 본론

2 - 1. 개요

마이크로 인스트럭션 포맷은 크게 수평 구조 또는 수직 구조로 설계될 수 있는데 각각의 방법에는 장단점이 존재한다. 수직 구조 마이크로 인스트럭션은 한 마이크로 사이클에 한 개의 마이크로 오퍼레이션이 수행되므로, 마이크로 인스트럭션의 길이는 줄일 수 있으나 마이크로 오퍼레이션의 병렬수행이 안된다. 그러나, 수평 구조 마이크로 인스트럭션은 한 마이크로 인스트럭션의 길이는 길어진다[9].

본 연구에서는 TM인 IBM/370 model 138이 고성능을 가진 범용 컴퓨터이므로 가능한 한 최대의 병렬 수행이 가능한 수평 구조 마이크로 인스트럭션 포맷을 설계하였다.

TM을 에뮬레이션하는 수평 구조 마이크로 인스트럭션 포맷의 설계 과정은 그림 1과 같으며 다음과 같이 설명된다.

1) TM분석

TM인 IBM/370의 컴퓨터 구조와 명령어 집합을 분석하여 논리적인 자원들을 정의한다.

2) 자원의 맵핑

정의된 논리적인 자원들에 대해 하드웨어 또는 firmware로의 구현 범위를 정의하여 1)번 과정에서 정의된 TM의 논리적인 자원들을 host machine의 하드웨어자원으로 자원맵핑을 한다.

3) 마이크로 오퍼레이션(MO) 정의

host machine이 정의되면 이것을 제어하기 위한 기본동작인 연산 MO, 전달 MO, 제어 MO 등을 정의한다.

4) 마이크로 오퍼레이션의 분류

정의된 MO들에 대해 상호 독립된 부분 집합으로 모은다.

5) 설 계

부분 집합으로 형성된 상호 독립적인 마이크로 오퍼레이션들을 필드 효율성, MO의 병렬 수행의 설계 기준을 고려하여 마이크로 인스트럭션 포맷을 형성한다.

6) 평 가

설계된 마이크로 인스트럭션 포맷에 대해 필드효율성과 MO의 병렬 수행 관점에서 그 결과가 만족되지 않을 경우에 host machine의 변경이 요구되면 2) 번 과정으로 또는 host의 변경이 요구되지 않으면 4) 번 과정으로 가서 개선 수정되어야 할 요인들을 반영하여 재설계한다.

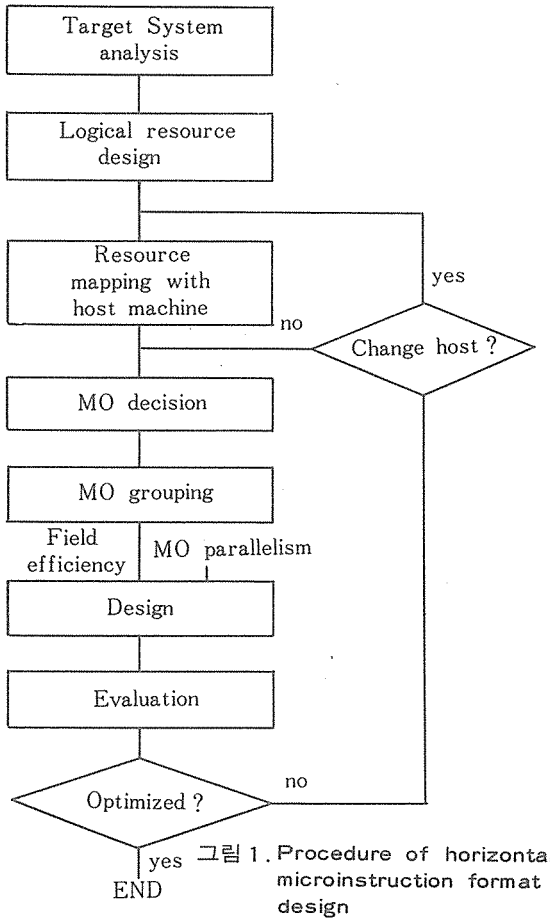


그림 1. Procedure of horizontal microinstruction format design

2 - 2. Target machine 분석

그림 2는 TM의 논리적인 블럭 다이어그램을 나타낸 것이며, 다음은 TM의 컴퓨터 구조와 명령어 집합이 가지는 논리적인 자원들을 정의한 것이다[10, 11, 12, 13, 14, 15].

1) GPR : 범용 레지스터

32비트의 GPR이 16개 존재하며 베이스 및 인덱스 레지스터는 이들 중에서 선택되어 사용된다.

2) FPR : 부동점 수 연산 레지스터

64비트의 FPR이 4개 존재하며 부동점 수 연산에 이용된다.

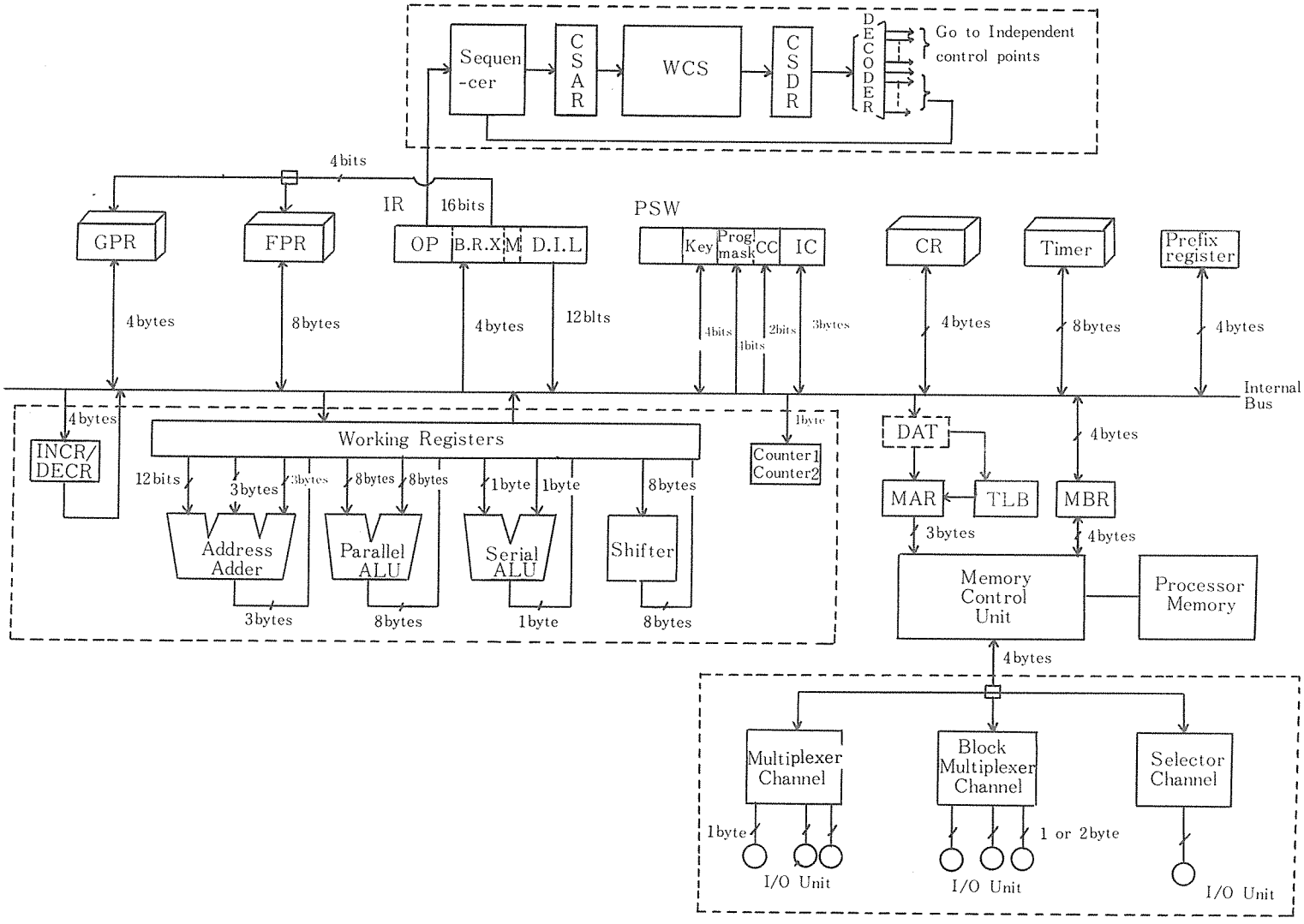


그림 2. IBM/370 logical block diagram

3) IR : 명령어 레지스터

IR은 현재 수행중인 명령어를 저장하는 32비트 레지스터이다. IR은 명령어 포맷에 따라 여러개의 필드로 나누어진다.

4) PSW : 프로그램 상태 레지스터

현재 수행하고 있는 명령어의 상태와 제어 정보 그리고 명령어 카운터를 갖는 레지스터로 그 크기는 64비트이다.

5) CR : 제어 레지스터

시스템의 제어 정보를 갖는 16개의 32비트 레지스터이며 가상 주소 번역, 채널 제어, 프로그램 수행 기록, 기계 점검 처리 등에 관한 정보를 갖고 있다.

6) TM : 타이머

현재 시각, 시각 비교, CPU 타이머 등을 갖고 있다.

7) IDR : 증가 / 감소 레지스터

명령어 카운터의 증가 등 여러 용도로 사용된다.

8) WR : 작업 레지스터

유효주소 계산과 연산의 자료 및 중간 계산 결과를 일시적으로 보관하는 버퍼이다.

9) AADD : 주소 가산기

베이스 레지스터의 내용과 인덱스 레지스터의 내용으로부터 유효주소 계산에 이용되는 가산기이다.

10) PALU : 병렬ALU

이진 연산과 부동점 수 연산중 mantissa 부분의 연산에 이용된다.

11) SALU : 순차적 ALU

부동점 수 연산 중 지수 부분 연산 및 1바이트씩 처리하는 십진 연산에 이용된다.

12) SHT : 쉬프트

산술적 및 논리적 쉬프트, 그리고 부동점 수 연산에서 mantissa부분의 정규화에 이용된다.

13) CNT : 카운터

가변 오퍼랜드를 가지는 인스트럭션의 길이필드를 갖고 바이트를 세는데 이용된다.

14) TLB : 주소 번역용 버퍼

가상 주소를 실 주소로 번역하는데 이용되는 하드웨어 버퍼이다.

15) MAR : 기억 장치 주소 레지스터

기억장치에 접근할 때 절대 주소를 갖는 24비트 버퍼이다.

16) MBR : 기억장치 버퍼 레지스터

기억장치에 접근할 때 MAR이 가리키는 주소의 내용을 가지는 32비트 버퍼이다.

2 - 3. 자원 맵핑

본 연구에서 사용한 host machine의 블럭 다이어그램은 그림 3과 같다. 사용된 host machine은 시스템 버스로 표준 버스인 VERSA버스를 가지는데 IBM/370을 에뮬레이션할 CPU는 기능적으로 data, path, sequencer 및 WCS 그리고 VERSA 버스 인터페이스로 구성되어 있다. 다음은 host machine의 하드웨어 자원에 2 - 2절에서 분석한 TM인 IBM/370의 논리적인 자원을 맵핑시킨 결과이다.

1. Data Path

data path는 수치적 및 논리적 연산을 수행하는 부분으로 중요한 자원들은 다음과 같다.

1) ALU

Am 29203 bit slice processor 8개가 병렬로 연결되어, TM의 이진 연산, 부동점 수 연산 십진 연산, 유효 주소 계산 등에 이용된다.

2) SLU

쉬프트 논리장치로 TM의 쉬프트 연산과 회전 연산 및 부동점 수 연산의 정규화에 이용된다.

3) FILE

쌍 포트(dual-port)를 가진 RAM배열로 TM의 32비트 범용 레지스터 16개, 32비트 제어 레지스터 16개, 64비트 부동점 수 레지스터 4개, 32비트 작업 레지스터 32개를 포함한다.

4) IR

32비트로 TM의 현재 수행하고 있는 명령어를 가지고 있다.

5) Mask/Rotate

Am 29203의 R포트 입력에 대한 니블(nibble) 단위의 마스크와 바이트 단위의 회전 기능을 수행한다.

6) Counter

TM의 에뮬레이션에서 바이트를 세는 데에 이용되는 범용 카운터이다.

7) Timer

TM의 현재 시간, 시각 비교, CPU 타이머의 에뮬레이션에 이용된다.

8) PSW

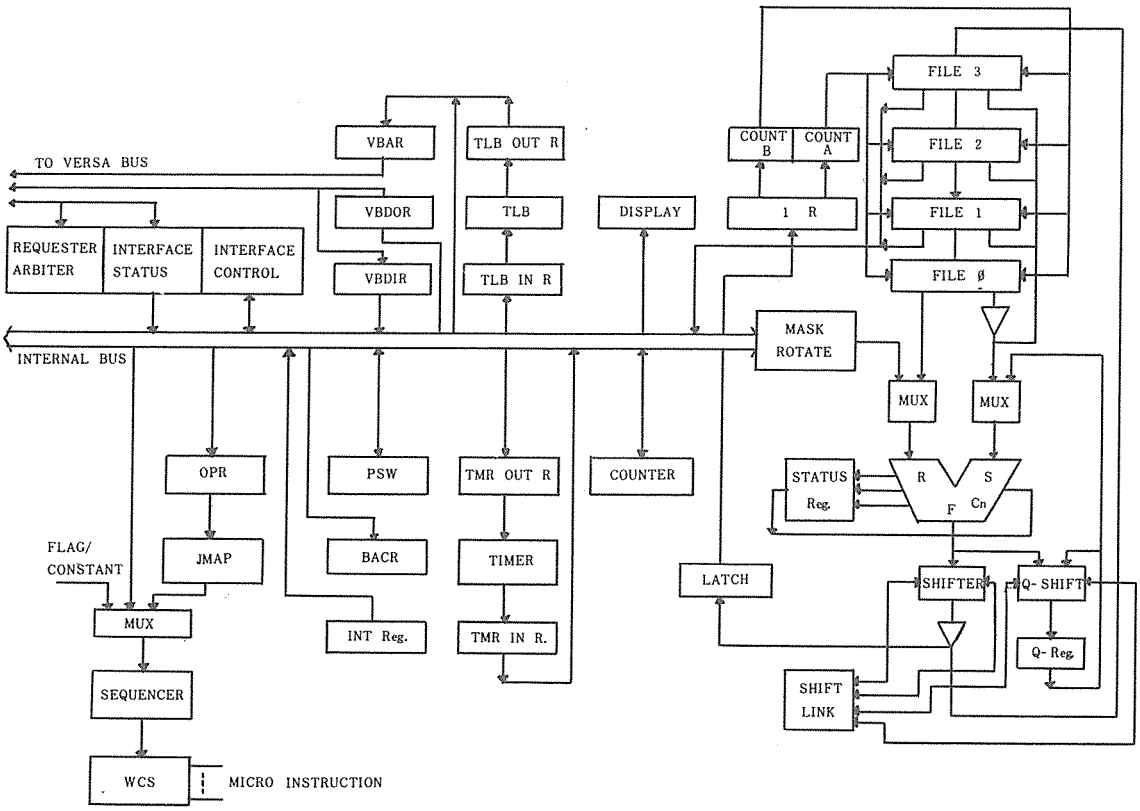


그림 3. Block diagram of host machine

TM의 64비트의 프로그램 상태 레지스터와 동일한 것으로 현재의 CPU상태를 나타낸다.

9) BACR

TM의 바이트 정렬 제어에 필요한 원시 주소, 목적 주소 및 바이트 갯수를 저장하는 레지스터이다.

10) INT

인터럽트 레지스터로 TM의 I/O요구와 외부 인터럽트 요구의 에뮬레이션에 이용된다.

11) Latch

Am 29203의 Y출력을 저장하는 latch이다.

2. Sequencer 및 WCS

마이크로 프로그램을 저장하고 수행하면서 시스템의 전반적인 제어를 하는 부분으로 그 자원들은 다음과 같다.

(1) Sequencer

AM2911 4개가 병렬로 연결되어 CPU 내에서 발생하는 모든 조건들을 고려하여 다음 수행할 마이크로 인스트럭션의 주소를 생성해 준다.

(2) WCS

80비트×8K워드로 TM명령어 집합을 에뮬레이션 할 마이크로 프로그램을 저장한다.

(3) Jmap

TM명령어의 OP코드 디코딩에 이용된다.

3. VERSA버스 인터페이스

버스 인터페이스는 버스 사양에 따라 data path와 VERSA 버스간의 인터페이스를 담당하는 것으로 주요 자원들은 다음과 같다.

(1) Requester/Arbiter

arbiter는 정해진 우선 순위에 따라 버스의 master를 결정하며, requester는 버스의 요구가 있을 경우 arbiter에게 버스를 요구하고 허락받은 다음 버스를 유지하는 기능을 가지고 있다.

(2) Master

입출력 레지스터의 제어와 slave를 선택하고 제어하기 위한 신호를 만든다.

(3) Slave

버스상의 주소와 주소 modifier의 내용을 해

독하여, 그것에 따라 필요한 기능을 수행한 후 master에게 응답을 보낸다.

(4) Interface status

조건 latch 및 오류 신호를 생성하는 부분으로 수행된 버스 사이클에 대한 정보를 제공하고 진단기능을 주는 플랙들이다.

(5) TLB

TM의 가상 주소를 실주소로 번역하는 데 이용되는 하드웨어 버퍼이다.

(6) VBAR

VERSA 버스 주소 레지스터이다.

(7) VBDIR

VERSA버스로부터 데이터를 받는 레지스터이다.

(8) VBDOR

VERSA 버스로 데이터를 내 보내는 레지스터이다.

2 - 4. 마이크로 오퍼레이션 분류

자원 맵핑을 한 후, host machine을 제어하는 MO를 정의하여 상호 독립적인 MO그룹을 형성하기 위해서 MO를 정의하였다. 정의된 MO는 그림 4 의 설계된 마이크로 인스트럭션 포맷에서 모든 요소의 집합과 같다. MO를 정의한 후 MO들을 상호 독립적인 부분 집합으로 모았는데 나누어진 큰 그룹들은 다음과 같다.

- AM29203의 R포트 입력제어
- 회전 제어
- AM29203의 기능 제어
- 캐리 제어
- 쉬프트 제어
- 다음 주소 제어
- 화일 쓰기 제어
- 범용 카운터 제어
- 주소 modifier 제어
- 상 수
- AM29203의 S포트 입력제어
- 마스크 제어
- 바이트 정렬 제어
- ALU 출력 저장 제어
- 조건 선택 제어
- 버스 요구 형태 제어
- 버스 R/W요구 제어
- 상태 코드 제어
- 디코드 형태 제어
- 직접 주소 값

2 - 5. 마이크로 인스트럭션 포맷 구성

2 - 4 절에서 모아진 MO그룹들을 다음의 과정에 의해 마이크로 인스트럭션 포맷을 구성하였다.

과정 1) MO들의 사용 빈도수와 마이크로 인스트럭션 포맷의 독립적인 필드 사이의 조화 관계를 고려하였다. 일반적으로는 마이크로 인스트럭션 포맷 설계에서 서로 상호 독립적인 MO들을 모아서 각 필드를 구성해 나가는 것이 최대 병렬 수행을 주는 방법이나, 실제적으로 MO의 사용 빈도수가 많지 않은데 그것들을 위해 독립적인 필드를 만들어 나간다면 마이크로 워드의 길이가 길어지고, 또 WCS의 크기도 크게 되어 시스템 전체 관점에서 볼 때 마이크로 인스트럭션의 각 필드들의 낭비가 많게 된다. 따라서 상호독립적인 MO그룹들의 빈도수를 고려하여 TM에뮬레이션에서 이용 빈도수가 낮은 버스 요구 형태 제어, 버스 R/W요구 제어, 범용 카운터 제어, 상태 코드 제어, 주소 modifier 제어 등을 2 개의 필드로 구성하였다.

과정 2) firmware의 관점을 고려하여 가능한 한 관계있는 필드들을 모아서 마이크로 프로그램의 개발과 유지를 쉽게 하기 위하여 기능적으로 서로 관계있는 필드를 모아서 기능별 그룹을 구성하였다. MO그룹들 중 AM29203의 기능 제어, 바이트 정렬 제어, 캐리 제어, 회전 제어 및 마스크 제어그룹을 모아 수치적 및 논리적 연산 제어 그룹으로 하고, AM29203의 R 포트 입력제어, AM 29203의 S포트 입력 제어, ALU 출력 저장 제어, 쉬프트 제어 및 화일 쓰기 제어 그룹을 모아 데이터 전달 제어 그룹으로 하였으며, 조건 선택 제어 및 다음 주소 제어 그룹으로 상수, 직접 주소값 및 디코드 형태 제어 그룹을 상수 그룹으로 그리고 과정 1에서 모은 2 개의 필드를 하드웨어 상태 제어 및 I/O 제어 그룹으로 구성하였다.

과정 3) 마이크로 프로그램 tool의 제약점을 고려하였다. 본 연구에서는 TM인 IBM/370의 에뮬레이션을 니모닉(mnemonic)으로 할 수 있게 하기 위해서 과정 2에서 모아진 각 기능별 그룹들을 AMD사에서 개발된 meta-assembler [16, 17]의 정의과정에서 다음과 같은 문법으로 200여개의 마이크로 인스트럭션을 정의해 보았다.

- 바이트 정렬 제어 그룹을 사용하지 않을 때
  S & D & N & A
- 바이트 정렬 제어 그룹을 사용할 때
  BA & N & A

여기서, S는 ALU기능 및 입력 제어 그룹이고, D는 ALU 출력 저장 제어 그룹이며, N은 다음 주소 제어 그룹이고, A는 보조 제어 그룹

이며, BA는 바이트 정렬 제어 그룹이고, &는 각 그룹들의 논리적인 OR이다.

위의 정의 과정에서 마이크로 프로그램의 효율성을 높이기 위해 회전 제어와 마스크 제어 부분을 수치적 및 논리적 연산 그룹에서 상수 그룹 뒤로 분리시켜서 그림 4와 같은 76비트, 18필드의 마이크로 인스트럭션 포맷을 설계하였다.

2-6. 설계된 마이크로 인스트럭션 포맷

그림 4의 설계된 마이크로 인스트럭션 포맷은 총 76비트, 18필드를 가지는 다 단계 (poly-phase)의 수평 구조 마이크로 인스트럭션 포맷으로 4개의 기능별 그룹을 가지는데, 각 그룹의 기능은 다음과 같다.

1) 수치적 및 논리적 연산 제어
(BA, AF, CN, RT 필드)

그림 4에서 BA필드가 0이 선택될 때 AF필드는 Am 29203 bit slice processor의 기능을 선택하는 부분이며, CN필드는 캐리 제어 부분이다. 그리고 BA 필드가 1이 선택될 때 AF필드는 바이트 정렬을 제어하는 부분으로 IBM 370 인스트럭션 중 십진 또는 문자열 데이터가 주기억 장치에 워드 단위로 구성되어 있지 않을 때, 이 데이터들을 주기억 장치로부터 효율적으로 읽고, 쓰는 오퍼레이션을 제어하는 부분이다. 그리고 RT필드와 C2필드에 있는 마스크 제어 부분은 ALU의 R포트 입력에 대해 1바이트 회전과 니블 단위의 마스크를 제어하는 부분으로 기능상으로는 ALU제어 부분과 같은 그룹에 속하나 본 연구에서 사용한 meta assembler의 positional argument특성 때문에 ALU제어부분과 분리되어 있다.

2) 데이터 전달 제어(RS, R, SC, SS, S, B, OM, SF 필드)

이 부분은 ALU의 입력 자원의 선택과 출력을 저장할 곳을 결정하는 부분으로 RS와 R필드는 ALU의 R포트 입력을 선택하고, SC, SS 및 S 필드는 ALU의 S포트의 입력을 선택한다. 그리고 B필드는 ALU의 출력을 화일 자원으로서 저장을 선택하는 부분으로 B필드가 1이면 SC, SS, S필드 등은 ALU 출력 저장 제어로 이용된다. OM필드는 ALU의 출력을 화일이 아닌 다른 자원으로의 저장할 곳을 선택하며, SF 필드는 ALU의 출력에 대해 쉬프트 연산을 제어한다.

3) 주소 제어(NC, CS 필드)

현재 수행하고 있는 마이크로 인스트럭션을 MI (n)이라 할때 MI(n-1) 수행 후의 CPU상태에 따라 다음에 수행할 마이크로 인스트럭션의 주소를 결정하는 부분으로 CS필드는 조건의 선택부분이고, NC 필드는 조건의 참과 거짓 상태에 따라 주소를 결정하는 주소 제어 선택부분이다.

4) 상수(CM, CL 필드)

CM과 CL필드는 자료 자신을 나타내는 부분으로, ALU의 입력 데이터, 다음 수행할 주소, 마스크 값, 분기 맵의 맵 형태 등으로 이용된

5) 하드웨어 상태 제어 및 I/O제어
(C1, C2 필드)

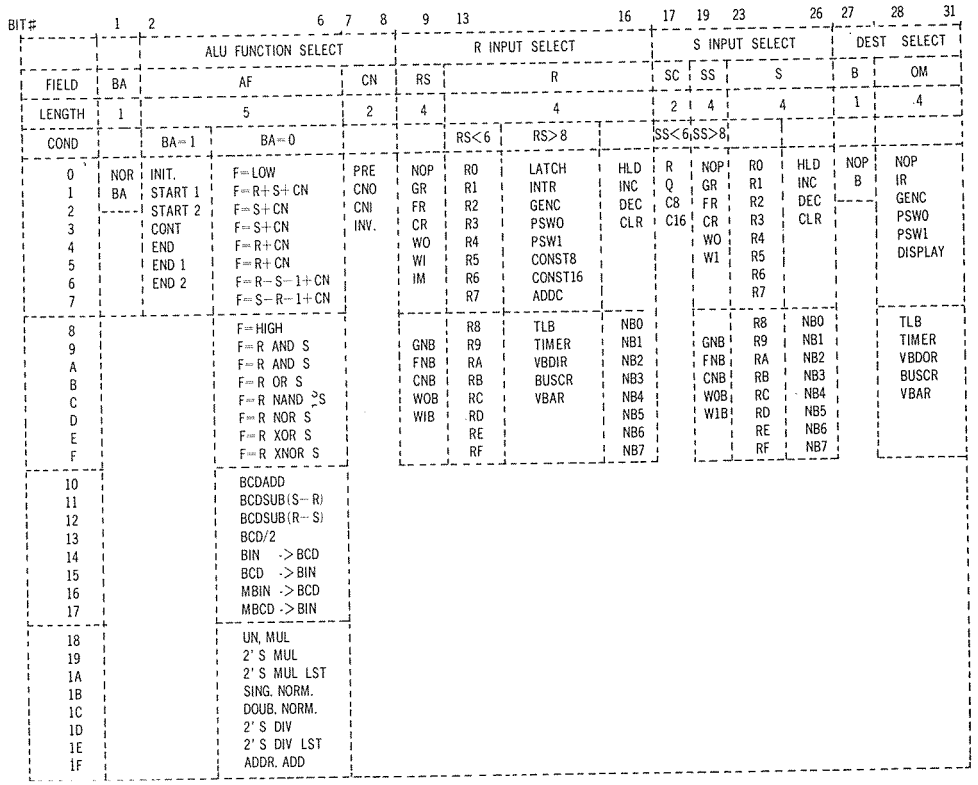
C1과 C2필드는 VERSA 버스 R/W 요구, 상태 코드, VERSA 버스 요구시의 주소 modifier, 범용 플랙, 타이머 등을 제어하는 부분으로 각 요소들은 서로 독립적인 오퍼레이션을 수행할 수 있으나 TM에뮬레이션에서 사용 빈도수가 적은 이유로 보조 제어 그룹의 2개의 필드로 모아져 있다.

3. 결론

디지탈 컴퓨터의 제어에 마이크로 프로그램에 의한 제어 방법이 확대됨에 따라 체계적인 마이크로 인스트럭션 포맷의 설계 방법이 요구되고 있다.

본 논문에서는 IBM/370을 에뮬레이션하기 위해서, target machine인 IBM/370을 분석하여 논리적인 자원을 정의하고, 정의된 논리적인 자원들을 host machine의 하드웨어 자원에 자원 맵핑을 시켜, 이 host machine을 효율적으로 제어하는 수평 제어 마이크로 인스트럭션 포맷을 설계하였다. 설계된 마이크로 인스트럭션 포맷은 meta-assembler를 이용하여 니모닉을 정의하고 마이크로 프로그램을 작성해 본 결과, TM인 IBM/370 명령어들을 에뮬레이션 할 수 있었다.

그러나, 설계 과정에서 마이크로 인스트럭션 포맷 설계에는 하드웨어 제약점이 많아서, 마이



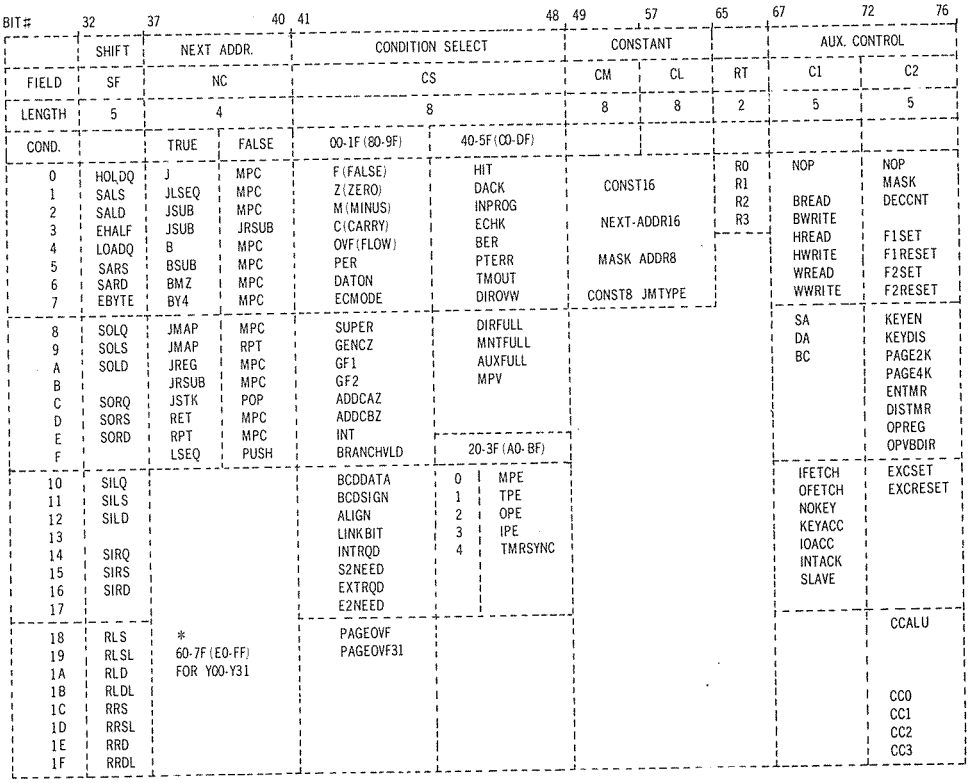


그림 4. Designed microinstruction format

크로 오퍼레이션의 병렬 수행 및 firmware엔지니어링 관점의 설계 기준들을 formal하게는 고려하지 못했다. 앞으로 이와 같은 설계 기준들을 formal하게 만족하는 설계방법에 대해 연구한다면 host machine을 최적으로 제어할 수 있는 마이크로 인스트럭션 포맷을 설계할 수 있을 것이다.

References

1) M. V. Wilkes, "The best way to design an automatic calculating machine," Report of the Manchester University Computer Inaugral Conference, Electrical Engineering Department of Manchester University, Manchester, England, July 1951.

2) Tomlinson G. Rausher and Phillip M. ADAMS, "Microprogramming : A tutorial and survey of recent developments," IEEE Tr. on Computer, Vol C-29, No. 1, Jan. 1980.

3) Takanobu Baba, "Current status and future directions of microprogramming," Micro. 16 tutorial, Oct. 1983.

4) Dilip K. Banerji and Jacques Raymond, Elements of microprogramming, Prentice-Hall, 1982.

5) Michal Andrews, Principles of firmware engineering in microprogram control, Computer Science Press, 1980.

6) Samir S. Husson, Microprogramming principles, Prentice-Hall, 1970.

7) George D. Kraft and Wing N. Toy, Microprogrammed control and reliable design of small computer, Prentice-Hall, 1981.

8) 과학기술처, "범용 컴퓨터 개발 중 32-bit 컴퓨터 구조 연구," 연구보고서, 1984. 3.

9) Subrata Dasgupta, "The organization of microprogram stores," Computing Surveys, Vol 11, No. 1, Mar. 1979.

10) Richard P. Case and Andris Padegs, "Architecture of IBM system/370," CACM Vol. 21, No. 1, pp. 73-96, Jan. 1978.

11) IBM, System/370 principles of operation, 1981.

12) IBM, A guide of the IBM system/370 model 138, 1978.

13) IBM, IBM system/370 model 138 functional characteristics, 1978.

14) A. Padegs, "System/360 and beyond," IBM J. Res. Develop. Vol. 25 No. 5, Sep. 1981.

15) Daniel P. Siewiorek, C. Gorden Bell and Allen Newell, Computer structure:principles and examples, McGraw-Hill, 1982.

16) Advaned Micro Computer, AMDASM29 manual, 1978.

17) Formation, Specification and user's manual for MPU-29 microassembler, 1980.