論文 87-24-2-8

# 마이크로프로그램의 레지스터 할당을 위한 변수결합 알고리즘

# (A Compatible Variables Scheduling Algorithm for Register Allocation in Microprogram)

李 相 靜*, 林 寅 七*

(Sang-Jeong Lee, In-Chil Lim)

**要 約**

본 논문에서는 마이크로프로그램의 각 변수들을 컴퓨터 시스템의 각 레지스터에 효율적으로 할당하기 위해 프로그램의 의미를 변경시키지 않으면서 같은 레지스터에 놓일 수 있는 결합변수군(compatible group)을 구하는 변수결합 알고리즘을 제안한다.

마이크로프로그램의 각 텍스트를 조사하여 T-V(Text-Variable) 행렬을 구성하여, 결합불능변수들(incompatible variables)을 구하고, 결합우선순위를 부여하여 변수결합을 시도함으로써 최소의 결합변수군을 구한다.

또 본 알고리즘을 VAX-11/780 C언어로 실현하여 실제 마이크로프로그램의 예에 적용시킴으로써 알고리즘의 타당성을 입증한다.

**Abstract**

**This paper proposes a compatible variables scheduling algorithm, which is the process to pack variables into same register without modifying program semantics, for efficient register allocation of microprogram.**

**The algorithm constructs T-V matrix, obtains incompatible variable set and scheduling priority, and schedules compatible groups. By this algorithm, the number of compatible groups can be minimized.**

**The algorithm was implemented with C language on VAX-11/780 computer. By applying the algorithm to practical microprograms, the effectiveness of the algorithm is verified.**

## I. 序 論

LSI/VLSI 기술의 발달로 컴퓨터 시스템이 복잡해지고, 시스템을 마이크로코드로써 제어하는 프로세서가 널리 사용됨에 따라 효율적인 마이크로코드를 자동생성하는 마이크로코드 자동생성 툴(automatic micro-code generation tool)에[1-5] 대한 요구가 증대되고 있는 실정이다.

특히, 마이크로코드화된 시스템에서는 주메모리(main memory)를 사용하는 비용이 많이 들기 때문에 효율적인 마이크로코드를 자동생성하기 위해선 마이크로프로그램의 각 변수들을 가능한 한 적은 수의 레지스터에 오래 머물도록 할당함으로써 주메모리 사용빈도를 줄이고자하는 레지스터 할당(register allocation) 알고리즘에 관한 연구가 활발히 진행중이다.[4-11]

이러한 연구중 DeWitt,[4] Ma[5], Mueller et. al.[11]등

*正會員, 漢陽大學校 電子工學科
(Dept. of Elec. Eng., Hanyang Univ.)

接受日字 : 1986年 9月 15日

은 오직 메모리와 레지스터 간의 이동만을 고려하여 최소화 시킴으로써 효율적인 레지스터 할당을 못한다는 단점이 있다.

Kim, Tan[8,9]은 메모리와 레지스터간의 이동외에 마이크로프로그램 내의 각 변수들에 대하여 프로그램의 의미를 변경시키지 않으면서 가능한 한 같은 레지스터에 놓일 수 있는 결합변수군을 구하여 최적 할당을 시도하였다.

결합변수군을 구하는 Kim, Tan의 알고리즘은 초기치로써 모든 변수들을 하나의 같은 결합변수군에 부가하고, 마이크로프로그램의 각 텍스트(text)를 조사하여 각 텍스트에서 현재 결합변수군에 할당된 변수들에 대하여 live 조건을 만족하지 않는 변수들을 제거함으로써 결합변수군들을 분리 생성하였다. 모든 텍스트에 대해 조사가 끝난 후 구해진 각 결합변수군들에 대해 prime implicant reduction 기법을 적용하여 최소의 결합변수군을 구함으로써 최적할당을 시도하였다.

그러나 Kim, Tan의 알고리즘은 흐름그래프(flow graph) 상의 모든 경로에 대하여 조사함으로써 중복된 기본블럭(basic block)의 각 텍스트에 대하여 중복조사를 하였다. 또한 prime implicant reduction 적용시 cyclic 문제가 발생하는 경우에 계산 시간이 너무 많이 걸리게 되어 리던던트(redundant)한 결합변수군이 구해질 수 있다는 문제점을 내포하고 있다.

본 논문에서는 이러한 문제점을 해결하기 위해서 마이크로프로그램의 각 텍스트와 변수를 조사하여 T-V 행렬을 구성하고 구성된 T-V행렬을 이용하여 결합불능변수 집합을 구하고, 결합우선순위를 부여하여 결합함으로써 prime implicant reduction 과정을 제거하여 계산시간을 줄임으로써 효율적으로 최소의 결합변수군을 구할 수 있는 변수결합 알고리즘을 제안한다.

또 본 알고리즘을 VAX-11/780 컴퓨터의 C언어로 실현하여 실제 마이크로프로그램에 적용시켜 봄으로써 본 알고리즘의 타당성을 보인다.

## Ⅱ. 變數結合 알고리즘

### 1. 텍스트 모델링 및 기본정의

먼저 본 알고리즘의 전개를 위해 필요한 텍스트 모델링 및 기본정의와 정리에 대해 서술한다.

본 알고리즘의 입력이 되는 마이크로프로그램을 머신 독립적인 중간언어 형태의 어셈블러 언어로 쓰여진 텍스트의 열(sequence)로 구성된다.

정의 1) 텍스트는 다음과 같이 6-tuple로 모델링한다.

〈TEXT#, INST, WR_V, RD_V1, RD_V2, NEXT_ADDR〉

여기서, TEXT# : 각 텍스트의 index

INST : 마이크로명령어

WR_V : WR(write)되는 변수

RD_V1, RD_V2 : RD(read) 되는 변수

NEXT_ADDR : 다음에 수행될 TEXT#

정의 2) 프로그램의 의미를 변경시키지 않으면서 같은 레지스터에 놓일 수 있는 마이크로프로그램 내의 각 변수들을 결합가능변수들(compatible variables)이라 하고 이 결합가능변수들의 집합을 결합변수군(compatible group)이라 한다.

정의 3) 임의의 변수와 같은 레지스터에 놓일 수 없는 변수들을 결합불능변수들(incompatible variables)이라 한다.

정의 4) 마이크로프로그램 상의 텍스트 t1, t2를 가정하고, t1에서 어떤 변수 v를 WR(write)하고 t1과 t2사이의 어떤 텍스트도 변수 v를 WR하지 않으면서 t2에서 v를 RD(read)하는 경우에 변수 v는 t1 이후의 텍스트에서부터 t2사이의 모든 텍스트에서 live 하다고 한다.

정의 5) 텍스트 t에서 변수v가 live하면서 RD하지 않는 경우에 변수v는 텍스트t에서 dormant live 하다고 한다.

정의 6) 1개의 시점 텍스트와 종점 텍스트를 가지며 그 사이에 어떤 분기(branch)문이나 레이블(label)문을 갖지 않는 텍스트의 열(sequence)을 기본블럭(basic block)이라 한다.

정의 7) 각 변수들에 대한 결합불능변수들의 수를 세어서 큰 순서대로 높은 결합우선순위(scheduling priority)를 부여한다. 결합불능변수의 수가 같은 경우에는 프로그램 상에 먼저 나타나는 변수에 높은 우선순위를 부여한다.

〔정리 1〕 어떤 변수의 결합확률은 결합불능변수의 수에 반비례한다.

(증명) 전체 변수의 집합V, 변수vx, vy의 결합불능변수 집합을 ICVX, ICVY라 하고, 결합불능변수의 수가 $|ICVX| > |ICVY|$ 라 하자.

변수 vx의 결합확률은

$P(vx) = (|V| - |ICVX|) / |V| = 1 - |ICVX| / |V|$

변수 vy의 결합확률은

$P(vy) = (|V| - |ICVY|) / |V| = 1 - |ICVY| / |V|$

그런데 $|ICVX| > |ICVY|$ 이므로

$P(vx) < P(vy)$ 가 된다.

따라서 어떤 변수의 결합불능변수의 수가 많으면 많을수록 다른 변수와 결합될 가능성은 적어진다.』QED

〔정리 2〕 어떤 변수를 먼저 결합시키면 다른 변수와 결합될 가능성은 커진다.

(증명) 전체 변수의 집합을 V, 어느 순간 t1에서의 결합될 변수의 집합을 V1, t1이후 t2에서의 결합될 변수의 집합을 V2라 하면 t1이후 t2이전에서 이미 결합된 변수들은 결합될 변수의 집합에서 제거되므로 |V1| > |V2| 가 된다.

또 임의 변수 vx의 결합불능변수의 집합을 ICVX라 했을 때, t1에서 vx가 결합될 확률

$P1(vx) = (|V1| - |ICVX|)/|V|$

t2에서 vx가 결합될 확률

$P2(vx) = (|V2| - |ICVX|)/|V|$

그런데 |V1| > |V2| 이므로, $P1(vx) > P2(vx)$ 가 된다. 따라서 어떤 변수를 먼저 결합시키는 것이 나중에 결합시키는 것보다 다른 변수와 결합될 확률이 커진다. 』QED

〔정리3〕 결합불능변수의 수가 큰 변수부터 먼저 결합시키는 것이 결합변수군을 줄일 수 있다.

(증명) 결합불능변수의 수가 크면 〔정리1〕에 의해 결합될 가능성이 적어지게 되고, 이러한 변수를 나중에 결합하면 〔정리2〕에 의해 결합가능성은 더욱 적어지게 되어 결합변수군의 수를 증가시키게 된다.

따라서 이러한 변수를 먼저 결합하여 결합 가능성을 높임으로써 결합변수군의 수를 줄일 수 있다. 』QED

2. 알고리즘

마이크로프로그램의 의미를 변경시키지 않으면서 마이크로프로그램 내의 각 변수들을 컴퓨터 시스템의 레지스터에 효율적으로 할당하기 위해 같은 레지스터에 놓일 수 있는 최소의 결합변수군을 구하는 본 알고리즘은 다음과 같은 과정으로 구성된다.

- 흐름그래프 구성
- live 변수 분석(live variable analysis)
- T-V행렬을 구성한다.
- WR-RD 텍스트 쌍을 구한다.
- 결합불능변수들의 집합을 구한다.
- 결합우선순위를 구한다.
- 결합변수군을 결정한다.

(1) 흐름그래프 구성

마이크로프로그램을 정의6)에 따라 기본블럭으로 나누고, 기본블럭을 절점(node)으로 하고 데이타 흐름을 에지(edge)로 하는 방향성 그래프(directed graph)인 흐름그래프를 구성한다.

(2) live변수 분석(live variable analysis)

구성된 흐름그래프에 있는 모든 기본블럭의 시점과 종점에서 live 한 변수들의 집합을 구하기 위해 다음과 같은 데이타흐름식(data flow equation)을 역방향(backward direction)으로 적용한다.

$OUT(b_i) = \cup IN(s)$

$IN(b_i) = [OUT(b_i) - WRV(b_i)] \cup RDV(b_i)$

여기서, $b_i$ : 기본블럭 i

s : $b_i$의 successors

$WRV(b_i)$ : 기본블럭 $b_i$에서의 WR 변수들의 집합

$RDV(b_i)$ : 기본블럭 $b_i$에서의 RD변수들의 집합
단, $b_i$에서 RD되기 전에 먼저 WR된 변수들은 제외한다.

$OUT(b_i)$ : 기본블럭 $b_i$의 종점에서 live 한 변수들의 집합

$IN(b_i)$ : 기본블럭 $b_i$의 시점에서 live 한 변수들의 집합

(3) T-V(Text-Variable) 행렬

마이크로프로그램의 흐름그래프 상의 각 경로(path)에 대하여 행에는 텍스트, 열에는 변수를 나타내는 2차원 행렬을 구성한다.

각 텍스트 상에서 RD, WR 되는 변수들을 조사하여 해당 행, 열에 각각 'R', 'W'로 표시하고, 동시에 RD, WR 되는 경우에는 'X'로 나타낸다. 또 각 텍스트에서 dormant live한 변수에 대해서는 'L'로 표시한다.

이러한 T-V행렬 구성시 프로그램의 의미를 보존하기 위해 다음 사항을 고려한다.

- 마이크로프로그램의 흐름그래프 상의 각 경로의 시점과 종점에서 가상 텍스트(dummy text)인 시점 텍스트(entry text), 종점 텍스트(exit text)를 각각 정의하고, 시점에서 live 한 변수들을 시점 텍스트에서의 WR 변수로 취급하고, 종점에서 live 한 변수들을 종점 텍스트에서의 RD 변수들로 취급한다.
- 각 루프(loop)의 시점에서 live 한 변수들을 각 루프의 마지막 텍스트에서의 RD 변수들로 취급한다.

(4) WR-RD 텍스트 쌍(WRTP)

각 경로의 T-V행렬에서 각 변수들을 WR, RD하는 텍스트를 조사하여 WR-RD텍스트 쌍의 집합을 구한다. 이때 RD 텍스트는 해당 WR 텍스트의 변수를 마지막으로 RD하는 텍스트이다.

$WRTP(v) = \{(tw_1, tr_1), (tw_2, tr_2), \cdots, (tw_k, tr_k), \cdots, (tw_n, tr_n)\}$

(단, $tw_i < tr_i <= tw_2 < \cdots <= tw_k < tr_k \cdots < tr_n$)

(5) 결합불능변수 집합(ICVS)

변수v의 결합불능변수들의 집합은 다음과 같이 구한다.

$$TEMP(k) = \bigcup_{t_i = tw_k + 1}^{tr_k} [RVS(t_i) \cup DLVS(t_i)]$$

$$ICVS(v) = \bigcup_{k=1}^{n} TEMP(k)$$

여기서, RVS($t_i$) : 텍스트 $t_i$에서의 RD 변수들의 집합
DLVS($t_i$) : 텍스트 $t_i$에서의 dormant live 한 변수들의 집합

(6) 결합우선순위

효율적인 레지스터 할당을 위해 최소의 결합변수군을 구하기 위해 정의7)과 같이 각 변수에 대하여 결합우선순위를 부여한다.

(7) 결합변수군 결정

마이크로프로그램 내의 모든 변수들의 집합을 V 라 했을 때 V=$\phi$가 될 때까지 다음 과정을 반복한다.

〔과정 1〕 결합변수군의 index i를 1로 초기화시킨다.

〔과정 2〕 결합불능변수들의 집합 CGIC=$\phi$로 놓고, 집합 V에서 결합 우선순위가 가장 높은 변수를 현재 조사되고 있는 변수vc에 할당하고, vc를 i번째 결합변수군, CG(i)에 할당한다.

〔과정 3〕 현재 조사되고 있는 CG(i)의 변수들과 결합시킬 수 없는 결합불능변수들의 집합 CGIC를 구한다.

CGIC=CGIC ∪ ICVS(vc)

〔과정 4〕 CG(i)의 변수들과 결합 가능한 결합가능변수들의 집합 CVS를 구한다.

CVS=V−CGIC

〔과정 5〕 CVS≠$\phi$인 경우, CVS에서 가장 결합우선순위가 높은 변수를 vc에 할당하고 vc를 CG(i)에 부가한 후 〔과정 3〕으로 간다.

〔과정 6〕 결합되지 않은 변수들의 집합V에서 CG(i)에 속한 변수들을 제거하고, i를 하나 증가시킨 후 V≠$\phi$이면 〔과정 2〕로 가고, V=$\phi$이면 과정을 멈춘다.

## Ⅲ. 適用 例 및 比較

이 장에서는 Kim, Tan이 제시한 예제 마이크로프로그램에 대해 Kim, Tan의 알고리즘과 본 알고리즘을 적용시켜 비교한다. 또 본 알고리즘을 VAX-11/780 C 언어로 실현하여 실제 마이크로프로그램 예들에 대해 적용시켜 비교, 검토한다.

그림 2의 예에서 변수 b1, b2와 e1, e2 그리고 f1, f2는 b, e, f를 서로 독립적인 변수로 분할한 것이다.

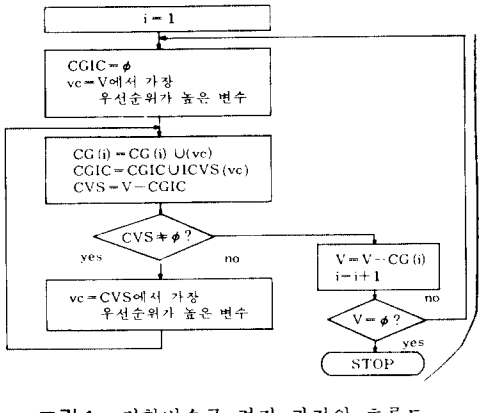


**그림 1.** 결합변수군 결정 과정의 흐름도
**Fig. 1.** Flow chart of compatible group scheduling procedure.

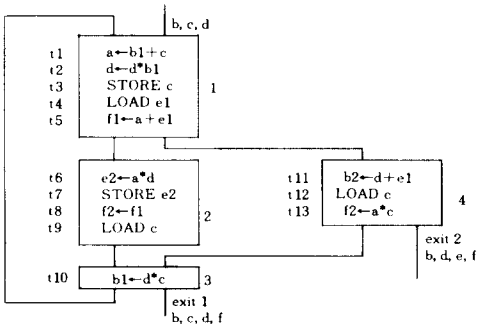


**그림 2.** Kim, Tan의 예제 마이크로프로그램의 흐름그래프
**Fig. 2.** Flow graph of Kim and Tan's example microprogram.

〔예제 1〕 그림 2의 예에 대한 Kim, Tan의 알고리즘 적용 예

그림 2의 경로 1 (t1−t10), 경로 2 (t1−t5, t11−t13, t−10)의 각 텍스트를 조사하여 다음과 같이 최대 결합변수군(maximal compatible group)을 구한다. 현재 조사되고 있는 텍스트의 WR변수를 vi, live한 변수들의 집합을 VJ라 했을 때 이전 텍스트에서 구해진 MG (maximal compatible group)들을 조사하여 MG중에서 vi와 VJ중 적어도 하나 이상의 변수를 가지고 있는 각 MG(c)에 대해 다음 식을 적용하여 새로운 MG, MG(i), MG(j)를 분리 생성한다.

MG(i)=MG(c)−VJ

MG(j)=MG(c)−{vi}

그림 2의 예에서 경로 1의 시점에서 초기치로 경로 1에 나타나는 모든 변수들을 MG에 할당하고, 경로 1의 각 텍스트에 대해 MG(i), MG(j)를 구한다. 경로 1에서 구해진 모든 MG들에 경로 2의 새로운 변수 b2를 부가하고 경로 2의 각 텍스트에 대해 MG(i), MG(j)를 구하면 아래와 같은 결과가 얻어진다.

MG(1)={a, e2, f2}, MG(2)={b1, b2, e2},
MG(3)={b1, b2, f1}, MG(4)={b1, e1, e2},
MG(5)={b1, e1, f1}, MG(6)={c, e2},
MG(7)={c, f1}, MG(8)={d},
MG(9)={f1, f2}

구해진 최대 결합변수군에 대해 모든 변수들을 포함하는 최소의 결합변수군을 구하기위해 prime implicant reduction을 적용시킨다(그림 3).

| 변수 \ MG | 1 ∨ | 2 | 3 | 4 | 5 | 6 | 7 | 8 ∨ | 9 |
|---|---|---|---|---|---|---|---|---|---|
| ∨ a | 1 | | | | | | | | |
| b1 | | 1 | 1 | 1 | 1 | | | | |
| b2 | | 1 | 1 | | | | | | |
| c | | | | | | 1 | 1 | | |
| ∨ d | | | | | | | | 1 | |
| e1 | | | | 1 | 1 | | | | |
| ∨ e2 | 1 | 1 | | 1 | | 1 | | | |
| f1 | | | 1 | | 1 | | 1 | | 1 |
| ∨ f2 | 1 | | | | | | | | 1 |

**그림 3.** 그림 2의 마이크로프로그램에 대한 prime implicant reduction
**Fig. 3.** Prime implicant reduction for Fig2's microprogram.

Kim, Tan은 prime implicant reduction 적용 시 branch and bound approach를 적용하여 각 변수를 cover할 수 있는 MG의 모든 가능성을 조사하고, 만약 계산시간이 어느 일정한 한계를 넘어서면 계산을 중단하고 리던던트한 결합변수군을 해로써 취하였다.

위의 예에서 MG(1), MG(8)만이 각각 a와 d를 포함하는 필수 결합변수군(essential compatible group)이므로 MG(1), MG(8)을 선택하고 이 결합변수군들에 포함되는 변수 a, d, e2, f2는 cover 문제에서 제외한다.

나머지 변수들에 대해서는 cover하는 MG의 선택순서에 따라 결합변수군의 수가 달라지는 cyclic 문제가 발생한다. 즉, 최선의 경우 b1, c, e1을 cover 하는 결합변수군으로 MG(3), MG(6), MG(4)를 선택하면, 나머지 변수 모두가 포함되어 모두 5개의 결합변수군이 구해지지만, 최악의 경우에는 b1, c, e1, f1을 cover하는 결합변수군으로 MG(2), MG(6), MG(4), MG(3)을 각각 선택하는 경우도 발생하여 모두 6개의 결합변수군이 선택된다.

〔예제 2〕 그림 2의 예에 대한 본 알고리즘의 적용예

먼저 각 기본블럭의 시점과 종점에서 live한 변수들을 다음과 같이 구한다.

IN(1)= {b1, c, d}, OUT(1)= {a, d, e1, f1},
IN(2)= {a, d, f1}, OUT(2)= {c, d, f2},
IN(3)= {c, d, f2}, OUT(3)= {b1, c, d, f2},
IN(4)= {a, d, e1}, OUT(4)= {b2, c, d, e1, f2},
OUT(exit1) = {b, c, d, f}, OUT(exit2) = {b, c, d, e, f}

Live 변수들을 구한 후 경로 1로써 t1-t10를 취하고 경로 2로써 나머지 텍스트 t11-t13를 취하여 경로 1, 2에 대해 각각 T-V행렬을 구성한다(그림 4). 이때 경로 1의 시점에서 b, c, d가 live 변수이므로 T-V행렬 구성 시 b1, c, d가 각각 tentry1에서 WR 변수가 된다. 또 b, c, d, f가 경로 1의 종점에서 live 변수이므로 texit1에서 RD 변수가 된다. 경로 2의 시점, 즉 기본블럭 4의 시점에서는 변수 a, d, e1이 live하므로 tentry2에서 WR변수가 되고, 종점(기본블럭 4의 종점) 에서는 b2, c, d, e1, f2가 live하므로 texit2에서 각각 RD 변수가 된다.

| 텍스트 \ 변수 | a | b1 | c | d | e1 | e2 | f1 | f2 |
|---|---|---|---|---|---|---|---|---|
| tentry1 | | W | W | W | | | | |
| t1 | W | R | R | L | | | | |
| t2 | L | R | L | X | | | | |
| t3 | L | | R | L | | | | |
| t4 | L | | | L | W | | | |
| t5 | R | | | L | R | | W | |
| t6 | R | | | R | | W | L | |
| t7 | | | | L | | R | L | |
| t8 | | | | L | | | R | W |
| t9 | | | W | L | | | | L |
| t10 | | W | R | R | | | | L |
| texit1 | | R | R | R | | | | R |

(a)

| 텍스트 \ 변수 | a | b2 | c | d | e1 | f2 |
|---|---|---|---|---|---|---|
| tentry2 | W | | | W | W | |
| t11 | L | W | | R | L | |
| t12 | L | L | W | L | L | |
| t13 | R | L | R | L | L | W |
| texit2 | | R | R | R | R | R |

(b)

**그림 4.** 그림 2 마이크로프로그램의 (a) 경로1, (b) 경로 2에 대한 T-V행렬
**Fig. 4.** T-V matrix for (a) path1 (b) path2 of Fig2's microprogram.

또한 각 변수들의 WRTP, ICVS는 다음과 같이 구해진다.

WRTP (a) = {(t1, t6, ) (tentry2, t13)},
WRTP (b1) = {(tentry1, t2), (t10, texit1)},
WRTP (b2) = {(t11, texit2)},

WRTP (c) = {(tentry1, t3), (t9, texit1), (t12, texit2)},
WRTP (d) = {(tentry1, t2), (t2, texit1), (tentry2, texit2)}
WRTP (e1) = {(t4, t5), (tentry2, texit2)},
WRTP (e2) = {(t6, t7)},
WRTP (f1) = {(t5, t8)},
WRTP (f2) = {(t8, texit1), (t13, texit2)}

ICVS (a) = {a, b1, b2, c, d, e1, f1},
ICVS (b1) = {a, b1, c, d, f2},
ICVS (b2) = {a, b2, c, d, e1, f2},
ICVS (c) = {a, b1, b2, c, d, e1, f2},
ICVS (d) = {a, b1, b2, c, d, e1, e2, f1, f2},
ICVS (e1) = {a, b2, c, d, e1, f2},
ICVS (e2) = {d, e2, f1},
ICVS (f1) = {a, d, e2, f1},
ICVS (f2) = {b1, b2, c, d, e1, f2}

ICVS 결과에 따라 결합우선순위를 구하고(그림 5), 각 변수의 우선순위대로 결합변수군을 구하면(그림6), 다음과 같이 모두 5개의 결합변수군이 구해진다.

{d}, {a, e2, f2}, {c, f1}, {b1, b2}, {e1}

부록에서는 머신 독립적인 중간언어 형태인 어셈블리 언어로 작성된 2진변환 마이크로프로그램을 적용시킨 결과를 나타냈다.

표 1은 모두 5개의 마이크로프로그램에 대해 본 알고리즘을 VAX-11/780 C언어로 실현하여 적용시킨 결과이다. 표 1에 나타난 바와 같이 평균 47% 정도의 결합률로써 결함됨을 알 수 있다.

## Ⅳ. 結 論

본 논문에서는 마이크로프로그램의 각 변수들을 컴퓨터 시스템의 각 레지스터에 효율적으로 할당하기 위해 마이크로프로그램의 의미를 변경시키지 않으면서 같은 레지스터에 놓일 수 있는 결합변수군을 구하는 변수결합 알고리즘을 제안하였다.

즉, 마이크로프로그램의 각 텍스트를 조사하여 T-V 행렬을 구성하여 각 변수들에 대한 결합불능변수 집합을 구한 후, 결합불능변수의 수에 따라 결합우선순위를 부여하여 우선순위가 높은 순서대로 결합함으로써 적은 계산시간으로 최소의 결합변수군을 구하였다.

또 본 알고리즘을 VAX-11/780 C언어로 실현하여 실제 마이크로프로그램에 적용시켜 봄으로써 본 알고

| 변 수 | ICVS의 수 | priority |
|---|---|---|
| a | 7 | 2 |
| b 1 | 5 | 7 |
| b 2 | 6 | 4 |
| c | 7 | 3 |
| d | 9 | 1 |
| e 1 | 6 | 5 |
| e 2 | 3 | 9 |
| f 1 | 4 | 8 |
| f 2 | 6 | 6 |

**그림 5**. 그림 2의 마이크로프로그램에 대한 결합우선순위
**Fig. 5**. Scheduling prioroty for Fig2's microprogram.

| | V | CGIC | CVS | Scheduling |
|---|---|---|---|---|
| CG(1) | a, b1, b2, c, d, e1, e2, f1, f2 | | | d |
| | | a, b1, b2, c, d, e1, e2, f1, f2 | φ | |
| CG(2) | a, b1, b2, c, e1, e2, f1, f2 | | | a |
| | | a, b1, b2, c, d, e1, f1 | e2, f2 | a, f2 |
| | | a, b1, b2, c, d, e1, f1, f2 | e2 | a, f2, e2 |
| | | a, b1, b2, c, d, e1, e2, f1, f2 | φ | |
| CG(3) | b1, b2, c, e1, f1 | | | c |
| | | a, b1, b2, c, d, e1, f2 | f1 | c, f1 |
| | | a, b1, b2, c, d, e1, e2, f1, f2 | φ | |
| CG(4) | b1, b2, e1 | | | b2 |
| | | a, b2, c, d, e1, f2 | b1 | b2, b1 |
| | | a, b1, b2, c, d, e1, f2 | φ | |
| CG(5) | e1 | | | e1 |
| | | a, b2, c, d, e1, f2 | φ | |
| | φ | | | |

**그림 6**. 그림 2의 마이크로프로그램에 대한 결합변수군 결정 과정
**Fig. 6**. Compatible group scheduling procedure for Fig2's microprogram.

**표 1**. 실제 마이크로프로그램에 대한 본 알고리즘의 적용 결과
**Table 1**. Result of application the algorithm to practical microprograms.

| | Kim, Tan의 예 | 2진변환 | 부동소숫점 덧 셈 | 부동소숫점 곱 셈 | bubble sort |
|---|---|---|---|---|---|
| 텍 스 트 수 | 13 | 25 | 57 | 36 | 20 |
| 기본블럭수 | 4 | 9 | 13 | 7 | 6 |
| 변 수 수 | 9 | 7 | 12 | 12 | 6 |
| 결합변수군수 | 5 | 5 | 8 | 6 | 5 |
| 결 합 률 | 56% | 43% | 42% | 59% | 33% |

리즘의 타당성을 입증하였다.

본 알고리즘을 데이타의 효율적인 주메모리 전송을 위한 spill 과정과 레지스터-레지스터 이동 과정에 결합 적용시킨다면 보다 효율적인 레지스터 할당이 예상된다.

## 부 록

2 진변환 마이크로프로그램의 예

| ⟨TEXT | #INST | WR_V | RD_V1 | RD_V2 | NEXT_ADDR⟩ |
|---|---|---|---|---|---|
| 1 | LOAD | XSOURCE | | | 2 |
| 2 | CLEAR | VAL | | | 3 |
| 3 | AND | SIGN | XSOURCE | '15' | 4 |
| 4 | STORE | | SIGN | | 5 |
| 5 | RSH | XSOURCE | XSOURCE | '4' | 6 |
| 6 | AND | DIGIT | XSOURCE | '15' | 7 |
| 7 | LOAD | ADDR | | | 8 |
| 8 | LOAD | PWR | ADDR | | 9 |
| 9 | INC | ADDR | ADDR | | 10 |
| 10 | STORE | | ADDR | | 11 |
| 11 | CLEAR | PPROD | | | 12 |
| 12 | CJMP | | DIGIT | 'Z' | 13, 14 |
| 13 | ADD | PPROD | PPROD | PWR | 14 |
| 14 | LSH | PWR | PWR | '1' | 15 |
| 15 | RSH | DIGIT | DIGIT | '1' | 16 |
| 16 | CJMP | | DIGIT | 'NZ' | 17, 12 |
| 17 | ADD | VAL | VAL | PPROD | 18 |
| 18 | RSH | XSOURCE | XSOURCE | '4' | 19 |
| 19 | CJMP | | XSOURCE | 'NZ' | 20, 6 |
| 20 | LOAD | SIGN | | | 21 |
| 21 | XOR | SIGN | SIGN | '11' | 22 |
| 22 | CJMP | | SIGN | 'NZ' | 23, 25 |
| 23 | NOT | VAL | VAL | | 24 |
| 24 | ADD | VAL | VAL | '1' | 25 |
| 25 | EXIT | | | | |

⟨basic block costitution⟩

basic block 1 → 1, 2, 3, 4, 5
left→ 2, right→
basic block 2 → 6, 7, 8, 9, 10, 11
left→ 3, right→
basic block 3 → 12
left→ 4, right→ 5
basic block 4 → 13
left→ 5, right→
basic block 5 → 14, 15, 16
left→ 6, right→ 3
basic block 6 → 17, 18, 19
left→ 7, right→ 2
basic block 7 → 20, 21, 22
left→ 8, right→ 9
basic block 8 → 23, 24
left→ 9, right→
basic block 9 → 25
left→ right→

⟨live variables at block-in, block-out⟩

block 1 → in →
out → VAL, XSOURCE
block 2 → in → VAL, XSOURCE
out → VAL, XSOURCE, PWR , DIGIT , PPROD
block 3 → in → VAL, XSOURCE, PWR , DIGIT , PPROD
out → VAL, XSOURCE, PWR , DIGIT , PPROD
block 4 → in → VAL, XSOURCE, PWR , DIGIT , PPROD
out → VAL, PPROD , XSOURCE, PWR, DIGI T
block 5 → in → VAL, PPROD , XSOURCE, PWR, DIGI T
out → VAL, PPROD , XSOURCE, PWR, DIGI T
block 6 → in → VAL, PPROD , XSOURCE
out → VAL, XSOURCE
block 7 → in → VAL
out → VAL
block 8 → in → VAL
out →
block 9 → in →
out →

⟨text-variable matrix⟩

| XSOURCE | VAL | SIGN | DIGIT | ADDR | PWR | PPROD |
|---|---|---|---|---|---|---|
| 0 | | | | | | |
| 1 W | | | | | | |
| 2 L | W | | | | | |
| 3 R | L | W | | | | |
| 4 L | L | R | | | | |
| 5 X | L | | | | | |
| 6 R | L | | W | | | |
| 7 L | L | | L | W | | |
| 8 L | L | | L | R | W | |
| 9 L | L | | L | X | L | |
| 10 L | L | | L | R | L | |
| 11 L | L | | L | | L | W |
| 12 L | L | | R | | L | L |
| 13 L | L | | L | | R | X |
| 14 L | L | | L | | X | L |
| 15 L | L | | X | | L | L |
| 16 R | R | | R | | R | R |
| 17 L | X | | | | | R |
| 18 X | L | | | | | |
| 19 R | R | | | | | |
| 20 | L | W | | | | |
| 21 | L | X | | | | |
| 22 | L | R | | | | |
| 23 | X | | | | | |
| 24 | X | | | | | |
| 25 | | | | | | |

⟨write-read text point pair⟩

variable XSOURCE → (1, 5), (5, 18), (18 19),
variable VAL → (2, 17), (17, 23), (23 24), (24, ),
variable SIGN → (3, 4), (20 21), (21 22),
variable DIGIT → (6, 15), (15 16),
variable ADDR → (7, 9), (9, 10),
variable PWR → (8, 14), (14 16),
variable PPROD → (11, 13), (13 17),

〈incompatible variable set〉

variable XSOURCE → XSOURCE, VAL, SIGN, DIGIT, ADDR, PWR, PPROD,
variable VAL → XSOURCE, VAL, SIGN, DIGIT, ADDR, PWR, PPROD,
variable SIGN → XSOURCE, VAL, SIGN,
variable DIGIT → XSOURCE, VAL, DIGIT, ADDR, PWR, PPROD,
variable ADDR → XSOURCE, VAL, DIGIT, ADDR, PWR,
variable PWR → XSOURCE, VAL, DIGIT, ADDR, PWR, PPROD,
variable PPROD → XSOURCE, VAL, DIGIT, PWR, PPROD,

〈scheduling priority〉

| variables | no. of icvs | priority |
|---|---|---|
| XSOURCE | 7 | 1 |
| VAL | 7 | 2 |
| SIGN | 3 | 7 |
| DIGIT | 6 | 3 |
| ADDR | 5 | 5 |
| PWR | 6 | 4 |
| PPROD | 5 | 6 |

〈compatible groups〉

CG 1 → XSOURCE,
CG 2 → VAL,
CG 3 → DIGIT, SIGN,
CG 4 → PWR,
CG 5 → ADDR, PPROD,

## 參 考 文 献


[1] R.J. Sheraga and J.L. Gieser, "Experiments in automatic microcode generation", *IEEE Trans. on Computer* vol. C-32, pp. 557-569, June 1983.

[2] S.R. Vegdahl, *Local Code Generation and Compaction in Optimizing Microcode Compliers.* Ph. D. thesis, Carnegie-Mellon Univ., 1982.

[3] P. Marwedel, "A retargetable compiler for a high-level microprogramming language", *IEEE 17th Annual Microprogramming Workshop*, pp. 267-274, Nov. 1984.

[4] D.J. DeWitt, *A machine Independent Approach to the Production of Horizontal Microcode.* Ph. D. thesis, Univ. of Michigan, 1976.

[5] P.Y. Ma, *Optimizing the Microcode Produced by a High Level Microprograming Language.* Ph. D thesis, Oregon Univ., 1979.

[6] J. Kim, C.J. Tan, *Register Assignment Algorithm-II.* IBM Research Report RC 6262 (#26848), Oct. 1976.

[7] J. Kim, *Sprill Placement Optimization in Register Allocation for Compilers.* IBM Research Report RC 7251 (#31233), Aug. 1978.

[8] C.J. Tan, *Register Assignment Algorithms for Optimizing Compilers.* IBM Research Report RC 6481 (#27981), Apr. 1977.

[9] J. Kim, C.J. Tan, *Register Assignment Algorithms for Optimizing Microcode Compilers.* IBM Research Report RC 7639 (#33035), May 1979.

[10] R.A. Mueller et al., "A strvey of resource allocation methods in optimizaing microcode compilers", *IEEE 17th Annual Microprogramming Workshop,* pp. 285-295, Nov. 1984.

[11] R.A. Mueller et. al., "Global methods in the flow graph approch to retargetable microcode generation", *IEEE 17th Annual Microprogramming Wrokshop,* pp. 275-284, Nov. 1984.

[12] A.V. Aho, J.D. Ullman, *Principles of Complier Design.* Addison-Wesley, 1977.

[13] 이상정, 문성일, 정하중, 임인칠, "마이크로프로그램의 레지스터 할당을 위한 Compatibility Scheduling 알고리즘," 大韓電子工學會 夏季綜合學術大會 論文集, vol. 9 no1, pp. 264-267, 1986년 6 월.