(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2024년10월11일
(11) 등록번호 10-2713738
(24) 등록일자 2024년09월30일

(51) 국제특허분류(Int. Cl.)
G06F 9/30 (2018.01) *G06F 9/38* (2024.01)
G06T 1/20 (2018.01)
(52) CPC특허분류
G06F 9/30141 (2013.01)
G06F 9/3012 (2013.01)
(21) 출원번호 10-2023-0110116
(22) 출원일자 2023년08월22일
심사청구일자 2023년08월22일
(56) 선행기술조사문헌
KR101940523 B1
KR1020160138878 A

(73) 특허권자
이화여자대학교 산학협력단
서울특별시 서대문구 이화여대길 52 (대현동, 이화여자대학교)
(72) 발명자
윤명국
서울특별시 종로구 통일로 246-20, 102동 603호 (무악동, 무악현대아파트)
정은비
서울특별시 서대문구 신촌로 169-6, 이대역 STARGATE 1607호 (대현동)
(74) 대리인
특허법인 무한

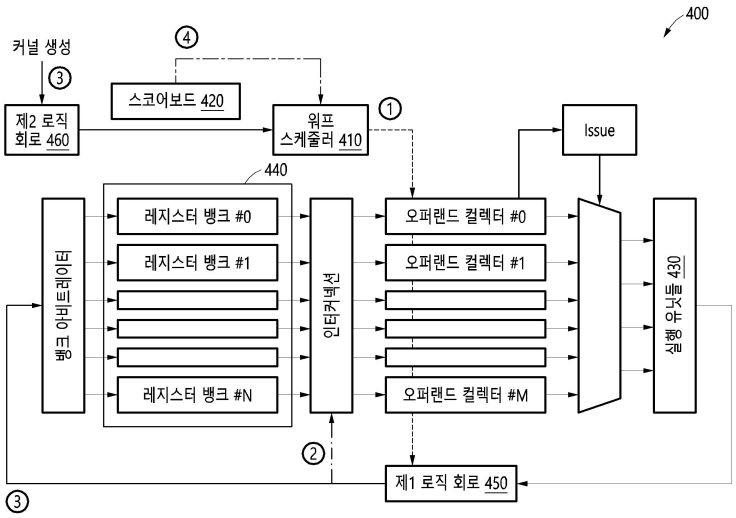
전체 청구항 수 : 총 17 항 심사관 : 지정훈

(54) 발명의 명칭 **오퍼랜드 컬렉터를 미리 할당하여 레지스터 파일을 효율적으로 사용하는 전자 장치 및 그 동작 방법**

(57) 요 약

본 개시는 오퍼랜드 컬렉터를 미리 할당하여 레지스터 파일을 효율적으로 사용하는 전자 장치 및 그 동작 방법에 관한 것이다. 본 개시의 일 실시예에 따른 전자 장치의 동작 방법은, 워프 스케줄러(warp scheduler)가 스코어보드(scoreboard)를 이용하여 다음 명령어에 대한 복수의 소스 오퍼랜드들(source operands) 중 하나 이상이 레지스터 파일(register file)에 저장되어 있는지 여부를 판단하는 동작, 상기 복수의 소스 오퍼랜드들 중 하나 이상이 레지스터 파일에 저장되어 있는 경우, 상기 다음 명령어에 대하여 오퍼랜드 컬렉터(operand collector)를 할당하는 동작, 이전 명령어의 연산 결과를 상기 오퍼랜드 컬렉터에 전송하는 제1 로직 회로를 이용하여 상기 연산 결과를 상기 오퍼랜드 컬렉터에 전송하는 동작, 및 상기 오퍼랜드 컬렉터에 로딩된 상기 연산 결과 및 상기 복수의 소스 오퍼랜드들 중 하나 이상을 이용하여 상기 다음 명령어에 대한 연산을 수행하는 동작을 포함할 수 있다.

대 표 도



(52) CPC특허분류
G06F 9/3836 (2023.08)
G06T 1/20 (2013.01)

이 발명을 지원한 국가연구개발사업
과제고유번호 1711179344
과제번호 00155966
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 인공지능융합혁신인재양성
연구과제명 인공지능융합혁신인재양성(이화여자대학교)
기 여 율 1/2
과제수행기관명 이화여자대학교산학협력단
연구기간 2023.01.01 ~ 2023.12.31
이 발명을 지원한 국가연구개발사업
과제고유번호 1711193622
과제번호 2021-0-02068-003
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 정보통신방송혁신인재양성
연구과제명 인공지능 혁신 허브 연구 개발
기 여 율 1/2
과제수행기관명 고려대학교산학협력단
연구기간 2023.01.01 ~ 2023.12.31

명 세 서

청구범위

청구항 1

전자 장치의 동작 방법에 있어서,

워프 스케줄러(warp scheduler)가 스코어보드(scoreboard)를 이용하여 다음 명령어에 대한 복수의 소스 오퍼랜드들(source operands) 중 하나 이상이 레지스터 파일(register file)에 저장되어 있는지 여부를 판단하는 동작;

상기 복수의 소스 오퍼랜드들 중 하나 이상이 레지스터 파일에 저장되어 있는 경우, 상기 다음 명령어에 대하여 오퍼랜드 컬렉터(operand collector)를 할당하는 동작;

이전 명령어의 연산 결과를 상기 오퍼랜드 컬렉터에 전송하는 제1 로직 회로를 이용하여 상기 연산 결과를 상기 오퍼랜드 컬렉터에 전송하는 동작; 및

상기 오퍼랜드 컬렉터에 로딩된 상기 연산 결과 및 상기 복수의 소스 오퍼랜드들 중 하나 이상을 이용하여 상기 다음 명령어에 대한 연산을 수행하는 동작

을 포함하는, 동작 방법.

청구항 2

제1항에 있어서,

상기 연산 결과를 전송 받기 전에 상기 복수의 소스 오퍼랜드들 중 하나 이상을 상기 레지스터 파일에서 상기 오퍼랜드 컬렉터로 미리 로딩하는 동작

을 더 포함하는, 동작 방법.

청구항 3

제1항에 있어서,

상기 오퍼랜드 컬렉터를 할당하는 동작은,

상기 오퍼랜드 컬렉터를 할당할 때 상기 연산 결과의 상기 오퍼랜드 컬렉터에 대한 포워딩 데이터를 상기 제1 로직 회로에 저장하는,

동작 방법.

청구항 4

제3항에 있어서,

상기 포워딩 데이터는,

상기 복수의 소스 오퍼랜드들 중 하나 이상을 제외한 나머지 소스 오퍼랜드의 식별자 데이터, 상기 오퍼랜드 컬렉터의 식별자 데이터 및 상기 연산 결과를 상기 레지스터 파일에 저장할 것 인지에 대한 데이터

를 포함하는, 동작 방법.

청구항 5

제4항에 있어서,

상기 연산 결과를 상기 레지스터 파일에 저장할 것 인지에 대한 데이터는,

데이터 의존성을 이용하여 상기 레지스터 파일에 상기 연산 결과를 저장할 것 인지 여부를 판단하는 제2 로직 회로의 출력으로 상기 워프 스케줄러를 통해 상기 제1 로직 회로에 전송되는,

동작 방법.

청구항 6

제4항에 있어서,

상기 연산 결과를 상기 오퍼랜드 컬렉터에 전송하는 동작은,

상기 포워딩 데이터의 상기 나머지 소스 오퍼랜드의 식별자 데이터와 상기 연산 결과에 대한 식별자 데이터를 비교하여 상기 연산 결과를 상기 오퍼랜드 컬렉터에 전송하는 동작

을 포함하는, 동작 방법.

청구항 7

제3항에 있어서,

상기 포워딩 데이터에 기반하여 상기 연산 결과를 상기 레지스터 파일에 저장할지 여부를 판단하는 동작

을 더 포함하는, 동작 방법.

청구항 8

제1항에 있어서,

상기 오퍼랜드 컬렉터를 할당하는 동작은,

상기 이전 명령어의 연산 시간이 임계값을 초과하는 경우, 상기 다음 명령어에 대하여 상기 오퍼랜드 컬렉터를 할당하지 않는,

동작 방법.

청구항 9

전자 장치의 동작 방법에 있어서,

워프 스케줄러(warp scheduler)가 스코어보드(scoreboard)를 이용하여 다음 명령어에 대한 복수의 소스 오퍼랜드들(source operands) 중 하나 이상이 레지스터 파일(register file)에 저장되어 있는지 여부를 판단하는 동작;

상기 복수의 소스 오퍼랜드들 중 하나 이상이 레지스터 파일에 저장되어 있는 경우, 상기 다음 명령어에 대하여 오퍼랜드 컬렉터(operand collector)를 할당하는 동작;

이전 명령어의 연산 결과를 상기 오퍼랜드 컬렉터에 전송하는 제1 로직 회로를 이용하여 상기 연산 결과를 상기 오퍼랜드 컬렉터에 전송하는 동작; 및

상기 제1 로직 회로에 저장된 포워딩 데이터에 기반하여 상기 연산 결과를 상기 레지스터 파일로 전송할지 여부를 판단하는 동작

을 포함하고,

상기 포워딩 데이터는,

데이터 의존성을 이용하여 상기 레지스터 파일로 상기 연산 결과의 저장을 판단하는 제2 로직 회로의 출력을 포함하는,

동작 방법.

청구항 10

명령어를 처리하는 프로세서를 포함하는 전자 장치에 있어서,

상기 프로세서는,

복수의 소스 오퍼랜드들을 저장하는 레지스터 파일;

스코어보드를 이용하여 다음 명령어에 대한 상기 복수의 소스 오퍼랜드들 중 하나 이상이 상기 레지스터 파일에 저장되어 있는지 여부를 판단하고 오퍼랜드 컬렉터를 상기 다음 명령어에 대해 할당하는 워프 스케줄러;

상기 다음 명령어에 대하여 할당되는 상기 오퍼랜드 컬렉터;

이전 명령어의 연산 결과를 상기 오퍼랜드 컬렉터에 전송하는 제1 로직 회로; 및

상기 오퍼랜드 컬렉터에 로딩된 상기 연산 결과 및 상기 복수의 소스 오퍼랜드들 중 하나 이상을 이용하여 상기 다음 명령어에 대한 연산을 수행하는 실행 유닛

을 포함하는, 전자 장치.

청구항 11

제10항에 있어서,

상기 오퍼랜드 컬렉터는,

상기 연산 결과를 전송 받기 전에 상기 복수의 소스 오퍼랜드들 중 하나 이상을 상기 레지스터 파일에서 상기 오퍼랜드 컬렉터로 로딩하는,

전자 장치.

청구항 12

제10항에 있어서,

상기 제1 로직 회로는,

상기 오퍼랜드 컬렉터를 할당할 때 상기 연산 결과의 상기 오퍼랜드 컬렉터에 대한 포워딩 데이터를 저장하는,

전자 장치.

청구항 13

제12항에 있어서,

상기 포워딩 데이터는,

상기 복수의 소스 오퍼랜드들 중 하나 이상을 제외한 나머지 소스 오퍼랜드의 식별자 데이터, 상기 오퍼랜드 컬렉터의 식별자 데이터 및 상기 연산 결과를 상기 레지스터 파일에 저장할 것 인지에 대한 데이터

를 포함하는, 전자 장치.

청구항 14

제13항에 있어서,

데이터 의존성을 이용하여 상기 레지스터 파일에 상기 연산 결과를 저장할 것 인지를 판단하는 제2 로직 회로를 더 포함하고,

상기 제2 로직 회로의 출력은,

상기 워프 스케줄러를 통해 상기 제1 로직 회로로 전송되는,

전자 장치.

청구항 15

제12항에 있어서,

상기 제1 로직 회로는,

상기 포워딩 데이터의 나머지 소스 오퍼랜드의 식별자 데이터와 상기 연산 결과에 대한 식별자 데이터를 비교하여 상기 연산 결과를 상기 오퍼랜드 컬렉터에 전송하는,

전자 장치.

청구항 16

제12항에 있어서,

상기 제1 로직 회로는,

상기 포워딩 데이터에 기반하여 상기 연산 결과를 상기 레지스터 파일에 저장할지 여부를 판단하는,

전자 장치.

청구항 17

제10항에 있어서,

상기 오퍼랜드 컬렉터는,

상기 이전 명령어의 연산 시간이 임계값을 초과하는 경우, 상기 다음 명령어에 대하여 할당되지 않는,

전자 장치.

발명의 설명

기 술 분 야

[0001] 본 개시는 오퍼랜드 컬렉터를 미리 할당하여 레지스터 파일을 효율적으로 사용하는 전자 장치 및 그 동작 방법에 관한 것이다.

배 경 기 술

[0002] GPU(graphics processing unit)는 다수의 스레드들(threads)를 동시에 처리할 수 있는 병렬성을 가질 수 있다. GPU는 그래픽 관련 애플리케이션뿐만 아니라 높은 연산 처리량을 요구하는 다양한 범용 애플리케이션에 사용될

수 있다. GPU는 스레드들 간의 빠른 문맥 교환(context switching)을 지원하기 위해서 큰 레지스터 파일(register file)을 사용할 수 있다. GPU의 전체 전력 소비량 중 20 % 이상은 레지스터 파일이 소비할 수 있다. 따라서, 레지스터 파일을 효율적으로 사용하기 위한 방법이 필요할 수 있다.

발명의 내용

해결하려는 과제

[0003] 본 개시는 명령어의 모든 소스 오퍼랜드들이 준비되지 않아도 오퍼랜드 컬렉터를 미리 할당하는 전자 장치 및 동작 방법을 제공할 수 있다.

[0004] 본 개시는 명령어의 연산 결과가 이후에 불필요한 경우 레지스터 파일에 전송하지 않는 전자 장치 및 동작 방법을 제공할 수 있다.

과제의 해결 수단

[0005] 본 개시의 일 실시예에 따른 전자 장치의 동작 방법은, 워프 스케줄러(warp scheduler)가 스코어보드(scoreboard)를 이용하여 다음 명령어에 대한 복수의 소스 오퍼랜드들(source operands) 중 하나 이상이 레지스터 파일(register file)에 저장되어 있는지 여부를 판단하는 동작, 상기 복수의 소스 오퍼랜드들 중 하나 이상이 레지스터 파일에 저장되어 있는 경우, 상기 다음 명령어에 대하여 오퍼랜드 컬렉터(operand collector)를 할당하는 동작, 이전 명령어의 연산 결과를 상기 오퍼랜드 컬렉터에 전송하는 제1 로직 회로를 이용하여 상기 연산 결과를 상기 오퍼랜드 컬렉터에 전송하는 동작, 및 상기 오퍼랜드 컬렉터에 로딩된 상기 연산 결과 및 상기 복수의 소스 오퍼랜드들 중 하나 이상을 이용하여 상기 다음 명령어에 대한 연산을 수행하는 동작을 포함할 수 있다.

[0006] 상기 연산 결과를 전송 받기 전에 상기 복수의 소스 오퍼랜드들 중 하나 이상을 상기 레지스터 파일에서 상기 오퍼랜드 컬렉터로 미리 로딩하는 동작을 더 포함할 수 있다.

[0007] 상기 오퍼랜드 컬렉터를 할당하는 동작은, 상기 오퍼랜드 컬렉터를 할당할 때 상기 연산 결과의 상기 오퍼랜드 컬렉터에 대한 포워딩 데이터를 상기 제1 로직 회로에 저장할 수 있다.

[0008] 상기 포워딩 데이터는, 상기 복수의 소스 오퍼랜드들 중 하나 이상을 제외한 나머지 소스 오퍼랜드의 식별자 데이터, 상기 오퍼랜드 컬렉터의 식별자 데이터 및 상기 연산 결과를 상기 레지스터 파일에 저장할 것 인지에 대한 데이터를 포함할 수 있다.

[0009] 상기 연산 결과를 상기 레지스터 파일에 저장할 것 인지에 대한 데이터는, 데이터 의존성을 이용하여 상기 레지스터 파일에 상기 연산 결과를 저장할 것 인지 여부를 판단하는 제2 로직 회로의 출력으로 상기 워프 스케줄러를 통해 상기 제1 로직 회로에 전송될 수 있다.

[0010] 상기 연산 결과를 상기 오퍼랜드 컬렉터에 전송하는 동작은, 상기 포워딩 데이터의 상기 나머지 소스 오퍼랜드의 식별자 데이터와 상기 연산 결과에 대한 식별자 데이터를 비교하여 상기 연산 결과를 상기 오퍼랜드 컬렉터에 전송하는 동작을 포함할 수 있다.

[0011] 상기 포워딩 데이터에 기반하여 상기 연산 결과를 상기 레지스터 파일에 저장할지 여부를 판단하는 동작을 더 포함할 수 있다.

[0012] 상기 오퍼랜드 컬렉터를 할당하는 동작은, 상기 이전 명령어의 연산 시간이 임계값을 초과하는 경우, 상기 다음 명령어에 대하여 상기 오퍼랜드 컬렉터를 할당하지 않을 수 있다.

[0013] 본 개시의 일 실시예에 따른 전자 장치의 동작 방법은, 워프 스케줄러(warp scheduler)가 스코어보드(scoreboard)를 이용하여 다음 명령어에 대한 복수의 소스 오퍼랜드들(source operands) 중 하나 이상이 레지스터 파일(register file)에 저장되어 있는지 여부를 판단하는 동작, 상기 복수의 소스 오퍼랜드들 중 하나 이상이 레지스터 파일에 저장되어 있는 경우, 상기 다음 명령어에 대하여 오퍼랜드 컬렉터(operand collector)를 할당하는 동작, 이전 명령어의 연산 결과를 상기 오퍼랜드 컬렉터에 전송하는 제1 로직 회로를 이용하여 상기 연산 결과를 상기 오퍼랜드 컬렉터에 전송하는 동작, 및 상기 제1 로직 회로에 저장된 포워딩 데이터에 기반하여 상기 연산 결과를 상기 레지스터 파일로 전송할지 여부를 판단하는 동작을 포함하고, 상기 포워딩 데이터는, 데이터 의존성을 이용하여 상기 레지스터 파일로 상기 연산 결과의 저장을 판단하는 제2 로직 회로의 출력을 포함할 수 있다.

[0014] 본 개시의 일 실시예에 따른 명령어를 처리하는 프로세서를 포함하는 전자 장치에 있어서, 상기 프로세서는, 복수의 소스 오퍼랜드들을 저장하는 레지스터 파일, 스코어보드를 이용하여 다음 명령어에 대한 상기 복수의 소스 오퍼랜드들 중 하나 이상이 상기 레지스터 파일에 저장되어 있는지 여부를 판단하고 오퍼랜드 컬렉터를 상기 다음 명령어에 대해 할당하는 워프 스케줄러, 상기 다음 명령어에 대하여 할당되는 상기 오퍼랜드 컬렉터, 이전 명령어의 연산 결과를 상기 오퍼랜드 컬렉터에 전송하는 제1 로직 회로, 및 상기 오퍼랜드 컬렉터에 로딩된 상기 연산 결과 및 상기 복수의 소스 오퍼랜드들 중 하나 이상을 이용하여 상기 다음 명령어에 대한 연산을 수행하는 실행 유닛을 포함할 수 있다.

[0015] 상기 오퍼랜드 컬렉터는, 상기 연산 결과를 전송 받기 전에 상기 복수의 소스 오퍼랜드들 중 하나 이상을 상기 레지스터 파일에서 상기 오퍼랜드 컬렉터로 로딩할 수 있다.

[0016] 상기 제1 로직 회로는, 상기 오퍼랜드 컬렉터를 할당할 때 상기 연산 결과의 상기 오퍼랜드 컬렉터에 대한 포워딩 데이터를 저장할 수 있다.

[0017] 상기 포워딩 데이터는, 상기 복수의 소스 오퍼랜드들 중 하나 이상을 제외한 나머지 소스 오퍼랜드의 식별자 데이터, 상기 오퍼랜드 컬렉터의 식별자 데이터 및 상기 연산 결과를 상기 레지스터 파일에 저장할 것 인지에 대한 데이터를 포함할 수 있다.

[0018] 데이터 의존성을 이용하여 상기 레지스터 파일에 상기 연산 결과를 저장할 것 인지를 판단하는 제2 로직 회로를 더 포함하고, 상기 제2 로직 회로의 출력은, 상기 워프 스케줄러를 통해 상기 제1 로직 회로로 전송될 수 있다.

[0019] 상기 제1 로직 회로는, 상기 포워딩 데이터의 상기 나머지 소스 오퍼랜드의 식별자 데이터와 상기 연산 결과에 대한 식별자 데이터를 비교하여 상기 연산 결과를 상기 오퍼랜드 컬렉터에 전송할 수 있다.

[0020] 상기 제1 로직 회로는, 상기 포워딩 데이터에 기반하여 상기 연산 결과를 상기 레지스터 파일에 저장할지 여부를 판단할 수 있다.

[0021] 상기 오퍼랜드 컬렉터는, 상기 이전 명령어의 연산 시간이 임계값을 초과하는 경우, 상기 다음 명령어에 대하여 할당되지 않을 수 있다.

발명의 효과

[0022] 본 개시의 일 실시예에 따르면, 명령어의 모든 소스 오퍼랜드들이 준비되지 않아도 오퍼랜드 컬렉터를 미리 할당하여 프로세서의 리소스를 효율적으로 활용할 수 있다.

[0023] 본 개시의 일 실시예에 따르면, 명령어의 연산 결과가 이후에 불필요한 경우 레지스터 파일에 전송하지 않음으로써 불필요한 쓰기 연산을 제거할 수 있다.

도면의 간단한 설명

[0024] 도 1은 본 개시의 일 실시예에 따른 전자 장치를 도시한 도면이다.

도 2는 오퍼랜드 컬렉터의 할당을 설명하기 위한 도면이다.

도 3은 데이터 의존성을 설명하기 위한 도면이다.

도 4는 본 개시의 일 실시예에 따른 오퍼랜드 컬렉터의 할당을 설명하기 위한 도면이다.

도 5는 본 개시의 일 실시예에 따른 제1 로직 회로를 도시한 도면이다.

도 6은 본 개시의 일 실시예에 따른 제2 로직 회로를 도시한 도면이다.

도 7은 본 개시의 일 실시예에 따른 전자 장치의 동작 방법을 설명하기 위한 플로우차트이다.

발명을 실시하기 위한 구체적인 내용

[0025] 이하, 실시예들을 첨부된 도면을 참조하여 상세하게 설명한다. 그러나, 특허출원의 범위가 이러한 실시예들에 의해 제한되거나 한정되는 것은 아니다. 각 도면에 제시된 동일한 참조 부호는 동일한 부재를 나타낸다.

[0026] 아래 설명하는 실시예들에는 다양한 변경이 가해질 수 있다. 아래 설명하는 실시예들은 실시 형태에 대해 한정하려는 것이 아니며, 이들에 대한 모든 변경, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다.

[0027] 제1 또는 제2 등의 용어는 다양한 구성요소들을 설명하는데 사용될 수 있지만, 이런 용어들은 하나의 구성요소를 다른 구성요소로부터 구별하는 목적으로만 이해되어야 한다. 예를 들어, 제1 구성요소는 제2 구성요소로 명명될 수 있고, 유사하게 제2 구성요소는 제1 구성요소로도 명명될 수 있다.

[0028] 실시예에서 사용한 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로, 실시예를 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수 개의 표현을 포함한다. 본 명세서에서, "A 또는 B", "A 및 B 중 적어도 하나", "A 또는 B 중 적어도 하나", "A, B 또는 C", "A, B 및 C 중 적어도 하나", 및 "A, B, 또는 C 중 적어도 하나"와 같은 문구들 각각은 그 문구들 중 해당하는 문구에 함께 나열된 항목들 중 어느 하나, 또는 그들의 모든 가능한 조합을 포함할 수 있다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 명세서 상에 기재된 특징, 숫자, 단계, 동작, 구성 요소, 부품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0029] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 실시예가 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가지고 있다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥 상 가지는 의미와 일치하는 의미를 가지는 것으로 해석되어야 하며, 본 출원에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.

[0030] 또한, 첨부 도면을 참조하여 설명함에 있어, 도면 부호에 관계없이 동일한 구성 요소는 동일한 참조 부호를 부여하고 이에 대한 중복되는 설명은 생략하기로 한다. 실시예를 설명함에 있어서 관련된 공지 기술에 대한 구체적인 설명이 실시예의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우 그 상세한 설명을 생략한다.

[0031] 이하, 본 발명의 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0032] 도 1은 본 개시의 일 실시예에 따른 전자 장치를 도시한 도면이다.

[0033] 도 1을 참조하면, 전자 장치(100)이 도시된다. 전자 장치(100)는 프로세서(110) 및 메모리(120)을 포함할 수 있다. 프로세서(110) 및 메모리(120)는 버스(bus), NoC(Network on a Chip), PCIe(Peripheral Component Interconnect Express) 등을 통하여 서로 통신할 수 있다. 도 1에 도시된 전자 장치(100)에는 본 실시예들과 관련된 구성요소들만이 도시되어 있다. 따라서, 전자 장치(100)는 도 1에 도시된 구성요소들 외에 다른 범용적인 구성요소들이 더 포함될 수 있음은 당업자에게 자명하다.

[0034] 프로세서(110)는 전자 장치(100)를 제어하기 위한 전반적인 기능을 수행하는 역할을 할 수 있다. 프로세서(110)는 메모리(120)에 저장된 프로그램들 및/또는 명령어들을 실행함으로써, 전자 장치(100)를 전반적으로 제어할 수 있다. 프로세서(110)는 전자 장치(100) 내에 구비된 CPU(central processing unit), GPU(graphics processing unit), AP(application processor) 등으로 구현될 수 있으나, 이에 제한되지 않는다.

[0035] 메모리(120)는 전자 장치(100) 내에서 처리된 데이터들 및 처리될 데이터들을 저장하는 하드웨어일 수 있다. 또한, 메모리(120)는 전자 장치(100)에 의해 구동될 애플리케이션, 드라이버 등을 저장할 수 있다. 메모리(120)는 DRAM(dynamic random access memory)과 같은 휘발성 메모리(volatile memory) 및/또는 불휘발성 메모리(nonvolatile memory)를 포함할 수 있다.

[0036] 아래에서 설명하는 프로세서(110)는 GPU로 구현될 수 있으나, 실시예가 이에 한정되지 않는다.

[0037] 프로세서(110)는 워프 스케줄러(warp scheduler)를 포함할 수 있다. 워프 스케줄러는 명령어의 모든 소스 오퍼랜드(source operand)들이 준비되었을 때 오퍼랜드 컬렉터(operand collector)를 할당할 수 있다. 소스 오퍼랜드는 명령어의 연산에 사용되는 피연산자일 수 있다. 다시 말해, 명령어의 모든 소스 오퍼랜드들이 레지스터 파일에 저장되면, 오퍼랜드 컬렉터가 할당될 수 있다. 오퍼랜드 컬렉터는 레지스터 파일에서 소스 오퍼랜드를 가져올 수 있다.

[0038] 상술한 오퍼랜드 컬렉터의 할당 방법은 모든 소스 오퍼랜드들이 레지스터 파일에 저장될 때까지 기다려야 할 수 있다. 따라서, 상술한 방법에 의하면 병목(bottleneck)에 의해서 프로세서(110)의 성능이 감소할 수 있다. 이하에서는 상술한 오퍼랜드 컬렉터를 할당하는 종래의 방법에 대해서 더 설명하도록 하겠다.

[0040] 도 2는 오퍼랜드 컬렉터의 할당을 설명하기 위한 도면이다.

[0041] 도 2를 참조하면, 프로세서(200)가 도시된다. 프로세서(200)는 워프 스케줄러(210), 스코어보드

(scoreboard)(220), 복수의 오퍼랜드 컬렉터(오퍼랜드 컬렉터 #0 내지 오퍼랜드 컬렉터 #M), 레지스터 파일(240) 및 복수의 실행 유닛들(230)을 포함할 수 있다. 레지스터 파일(240)은 복수의 레지스터 뱅크(레지스터 뱅크 #0 내지 레지스터 뱅크 #N)를 포함할 수 있다.

[0042] 워프 스케줄러(210)는 스코어보드(220)를 이용하여 명령어가 준비되었는지 여부를 확인할 수 있다. 스코어보드(220)는 소스 오퍼랜드의 준비 상태를 비지 비트(busy bit)로 저장할 수 있다. 각 레지스터 뱅크는 소스 오퍼랜드를 저장할 수 있다. 명령어에 대한 소스 오퍼랜드들 중 하나라도 준비되지 않은 경우, 해당 명령어는 준비되지 않은 상태로 판단될 수 있다. 명령어가 준비되지 않은 경우 오퍼랜드 컬렉터는 할당되지 않을 수 있다. 명령어가 준비된 경우, 오퍼랜드 컬렉터가 할당될 수 있다. 할당된 오퍼랜드 컬렉터는 명령어에 대한 소스 오퍼랜드들을 로딩할 수 있다. 복수의 실행 유닛들(230) 중 하나 이상의 실행 유닛은 로딩된 소스 오퍼랜드들을 이용하여 명령어에 따른 연산을 실행할 수 있다.

[0043] 상술한 방법으로 오퍼랜드 컬렉터를 할당하는 경우, 명령어에 대한 소스 오퍼랜드들 중 하나라도 이전 명령어(previous instruction)에 의한 연산 과정 중에 있는 경우, 해당 명령어는 오퍼랜드 컬렉터를 할당 받지 못하고 대기해야 할 수 있다. 즉, 오퍼랜드 컬렉터를 충분히 활용하지 못할 수 있다. 따라서, 명령어의 실행이 미루어질 수 있다. 마찬가지로, 레지스터 파일(240)도 효율적으로 사용되지 않을 수 있다. 오퍼랜드 컬렉터를 할당하는 단계는 전체 파이프라인(pipeline)의 수행 시간에서 25 % 이상을 차지할 수 있다.

[0044] 오퍼랜드 컬렉터를 할당하는 다른 방법으로 오퍼랜드 컬렉터 내부에 엔트리(entry)를 추가하여 데이터를 포워딩하는 확장된 오퍼랜드 컬렉터를 이용하는 방법이 있을 수 있다. 레지스터 파일(240)에서 확장된 오퍼랜드 컬렉터로 전달된 소스 오퍼랜드는 일정 범위내에 있는 다음 명령어(next instruction)로 포워딩 될 수 있다. 확장된 오퍼랜드 컬렉터를 이용하면 명령어의 레지스터 파일(240)에 대한 접근 대기 시간(access latency)이 감소할 수 있다. 또한, 반복적인 레지스터 파일(240)에 대한 쓰기 및/또는 읽기 연산을 줄여 전력 소모량을 감소시킬 수 있다.

[0045] 상술한 다른 방법은 오퍼랜드 컬렉터로 데이터를 로딩하는 시간을 줄일 수 있다. 또한, 레지스터 파일(240)에 대한 접근을 감소시킴으로써 레지스터 뱅크들을 효율적으로 사용할 수 있다. 다만, 하드웨어 오버헤드(hardware overhead)가 발생할 수 있으며, 프로세서(200)의 구성요소들과 오퍼랜드 컬렉터 사이의 인터커넥션(interconnection)이 복잡해지는 문제가 있을 수 있다.

[0046] 이하에서는 오퍼랜드 컬렉터를 명령어에 미리 할당하는 방법을 설명하기에 앞서 데이터 의존성에 대하여 설명하도록 하겠다.

[0048] 도 3은 데이터 의존성을 설명하기 위한 도면이다.

[0049] 프로세서는 명령어를 순차적으로 실행할 수 있다. 따라서, 명령어에 대한 어느 소스 오퍼랜드가 이전 명령어의 연산 결과를 이용하는 경우가 있을 수 있다. 이를 데이터 의존성(data dependency)라고 할 수 있다. 도 3을 참조하면, 일련의 순차적인 명령어들에서 데이터 의존성이 발생되는 예시가 도시된다.

[0050] 명령어 i4의 R9(9번 레지스터)에 해당하는 소스 오퍼랜드는 이전 명령어인 i3의 연산 결과에 해당할 수 있다. 명령어 i4는 이전 명령어인 i3의 연산 결과를 소스 오퍼랜드로 이용할 수 있다. 다시 말해, 데이터 의존성이 발생할 수 있다. 이때, 도 2에서 상술한 오퍼랜드 컬렉터를 할당하는 방식에 따르면, 명령어 i4는 R5(5번 레지스터)에 소스 오퍼랜드가 저장되어 있더라도, R9에 대한 연산이 종료한 후에 오퍼랜드 컬렉터를 할당 받을 수 있다.

[0051] R7(7번 레지스터)은 명령어 i4의 연산 결과가 저장된 레지스터로, 다음 명령어 i5에서 소스 오퍼랜드로 사용된 후에 다음 명령어 i5의 연산 결과에 의해 바로 업데이트될 수 있다. 이와 같이, 명령어의 연산 결과가 이후에 더 이상 다른 명령어의 소스 오퍼랜드로 사용되지 않거나 다음 명령어의 연산 결과에 의해 바로 업데이트 되는 경우, 명령어의 연산 결과를 레지스터 파일로 전달할 필요가 없을 수 있다. 다시 말해, 데이터 의존성이 발생했지만, 명령어의 연산 결과를 레지스터 파일로 전달할 필요가 없을 수 있다. 명령어의 연산 결과를 레지스터 파일로 전달할 필요가 있는지 여부는 컴파일 타임(compile time)에 분석할 수 있다.

[0052] 명령어에 대한 하나의 소스 오퍼랜드라도 레지스터 파일에 저장되어 있는 경우, 오퍼랜드 컬렉터를 미리 할당함으로써 레지스터 파일에 대한 접근 대기 시간을 줄일 수 있다. 예를 들어, 오퍼랜드 컬렉터를 미리 할당하여 레지스터 파일에 저장되어 있는 소스 오퍼랜드를 미리 가져오고 이전 명령어의 연산 결과를 오퍼랜드 컬렉터로 전달하면 대기 시간을 줄일 수 있다. 또한, 명령어의 연산 결과를 레지스터 파일로 전달할 필요가 없는 경우,

명령어의 연산 결과를 쓰는 연산을 제거함으로 레지스터 뱅크들을 효율적으로 사용할 수 있다.

[0053] 이하에서는 상술한 데이터 의존성을 기반으로 한 오퍼랜드를 할당하는 방법에 대해서 설명하도록 하겠다.

[0055] 도 4는 본 개시의 일 실시예에 따른 오퍼랜드 컬렉터의 할당을 설명하기 위한 도면이다.

[0056] 도 4를 참조하면, 프로세서(400)가 도시된다. 프로세서(400)는 워프 스케줄러(410), 스코어보드(scoreboard)(420), 복수의 오퍼랜드 컬렉터들(오퍼랜드 컬렉터 #0 내지 오퍼랜드 컬렉터 #M), 레지스터 파일(440), 복수의 실행 유닛들(430) 및 제1 로직 회로(450) 및 제2 로직 회로(460)를 포함할 수 있다. 레지스터 파일(440)은 복수의 레지스터 뱅크(레지스터 뱅크 #0 내지 레지스터 뱅크 #N)를 포함할 수 있다.

[0057] 이하에서는 오퍼랜드 컬렉터를 할당하는 방법을 동작 ① 내지 동작 ④를 이용하여 설명하도록 하겠다. 다만, 동작 ① 내지 동작 ④는 순차적으로 수행될 수도 있으나, 반드시 순차적으로 수행되는 것은 아니다. 예를 들어, 각 동작들의 순서가 변경될 수도 있으며, 적어도 두 동작들이 병렬적으로 수행될 수도 있다.

[0058] 동작 ①에서, 워프 스케줄러(410)는 스코어보드(420)를 이용하여 다음 명령어에 대한 복수의 소스 오퍼랜드들 중 하나 이상이 레지스터 파일(440)에 저장되어 있는지 여부를 판단할 수 있다. 워프 스케줄러(410)는 다음 명령어에 대한 복수의 소스 오퍼랜드들 중 하나 이상이 준비되어 있는지 여부를 확인할 수 있다.

[0059] 프로세서(400)는 명령어들을 순차적으로 처리할 수 있다. 따라서, 실행 유닛들(430) 중 어느 하나 이상에서 연산이 실행 중인 명령어를 이전 명령어라고 하면, 이전 명령어 이후에 수행될 명령어는 다음 명령어라고 할 수 있다. 이전 명령어와 다음 명령어는 연속적으로 수행되는 명령어들일 수 있다. 다만, 이는 예시일 뿐이고, 본 개시는 이에 한정되지 않는다.

[0060] 다음 명령어에 대한 복수의 소스 오퍼랜드들 중 하나 이상이 레지스터 파일(440)에 저장되어 있는 경우, 다음 명령어에 대한 오퍼랜드 컬렉터가 할당될 수 있다. 할당되는 오퍼랜드 컬렉터는 복수의 오퍼랜드 컬렉터들 중 임의의 하나일 수 있다.

[0061] 이때, 복수의 소스 오퍼랜드들 중 하나 이상을 제외한 나머지 소스 오퍼랜드의 식별자 데이터가 제1 로직 회로(450)에 저장될 수 있다. 오퍼랜드 컬렉터의 식별자 데이터(즉, 미리 할당된 오퍼랜드 컬렉터의 식별자 데이터)가 제1 로직 회로(450)에 저장될 수 있다. 이전 명령어의 연산 결과를 레지스터 파일(440)에 저장할 것인지에 대한 데이터가 제1 로직 회로(450)에 저장될 수 있다. 제1 로직 회로(450)에 저장된 데이터들은 이전 명령어의 연산이 종료되었을 때 이전 명령어의 연산 결과를 미리 할당된 오퍼랜드 컬렉터로 전송할 때 이용될 수 있다.

[0062] 동작 ②에서, 이전 명령어의 연산 결과는 레지스터 파일(440)로 전달되기 전에 제1 로직 회로(450)로 전달될 수 있다. 이전 명령어의 연산 결과가 데이터 의존성이 있는 경우, 제1 로직 회로(450)는 ①에서 할당된 오퍼랜드 컬렉터로 이전 명령어의 연산 결과가 전송될 수 있다. 이전 명령어의 연산 결과가 다음 명령어의 소스 오퍼랜드로 이용되는 경우, 제1 로직 회로(450)는 이전 명령어의 연산 결과를 다음 명령어의 오퍼랜드 컬렉터로 전송할 수 있다. 이전 명령어의 연산 결과가 데이터 의존성이 없는 경우, 이전 명령어의 연산 결과는 오퍼랜드 컬렉터로 전송되지 않고 레지스터 파일(440)로 전송될 수 있다.

[0063] 오퍼랜드 컬렉터를 미리 할당 받은 다음 명령어는 레지스터 파일에서 복수의 소스 오퍼랜드들 중 하나 이상을 오퍼랜드 컬렉터로 미리 로딩할 수 있다. 로딩된 복수의 소스 오퍼랜드들 중 하나 이상은 레지스터 파일에 저장되어 있는 다음 명령어에 대한 소스 오퍼랜드들 일 수 있다. 그러고나서, 이전 명령어의 연산 결과가 제1 로직 회로(450)를 통해 오퍼랜드 컬렉터로 포워딩 될 수 있다. 제1 로직 회로(450)에 대해서는 도 5에서 추가로 설명하도록 하겠다. 이전 명령어의 연산 결과가 복수의 소스 오퍼랜드들 중 하나 이상을 제외한 나머지 소스 오퍼랜드로 사용될 수 있다. 다음 명령어에 대한 모든 소스 오퍼랜드들이 로딩되었을 때 실행 유닛들(430) 중 하나 이상은 연산을 시작할 수 있다. 다음 명령어에 대한 모든 소스 오퍼랜드들은 오퍼랜드 컬렉터로 로딩된 이전 명령어의 연산 결과 및 레지스터에 저장된 복수의 소스 오퍼랜드들 중 하나 이상일 수 있다.

[0064] 일 실시예에 따르면, GPU의 명령어는 2개 이상의 소스 오퍼랜드를 필요로 할 수 있다. 따라서, 프로세서(400)가 GPU인 경우, 다음 명령어에 대한 2개의 소스 오퍼랜드들 중 하나가 레지스터 파일에 저장되어 있는 경우 오퍼랜드 컬렉터가 할당될 수 있다.

[0065] 동작 ③에서, 이전 명령어의 연산 결과는 제1 로직 회로(450)를 통해 오퍼랜드 컬렉터로 전송된 이후 다시 사용되지 않을 수 있다. 예를 들어, 이전 명령어의 연산 결과가 레지스터 파일에 저장되더라도 저장된 연산 결과가 다른 명령어의 소스 오퍼랜드로 이용되지 않으면 연산 결과는 다시 사용되지 않을 수 있다. 예를 들어, 이전

명령어의 연산 결과가 다음 명령어의 연산 결과에 의해서 바로 업데이트되는 경우 이전 명령어의 연산 결과는 다시 사용되지 않을 수 있다.

[0066] 이전 명령어의 연산 결과가 다시 사용되지 않는 경우, 이전 명령어의 연산 결과는 오퍼랜드 컬렉터에만 전송될 수 있다. 그리고, 이전 명령어의 연산 결과는 레지스터 파일에 전달되지 않을 수 있다. 다시 말해, 이전 명령어의 연산 결과를 레지스터 파일에 쓰는 연산이 제거될 수 있다. 이전 명령어의 연산 결과가 전송될 곳을 결정하기 위해 제2 로직 회로(460)가 이용될 수 있다. 제2 로직 회로(460)에 대해서는 도 6에서 더 설명하도록 하겠다.

[0067] 오퍼랜드 컬렉터를 다음 명령어에 대해 미리 할당하는 경우에도 다음 명령어의 실행이 지연될 수 있다. 이전 명령어의 긴 연산 시간에 때문에 긴 대기 시간을 갖는 다음 명령어는 오퍼랜드 컬렉터를 미리 할당 받더라도 이전 명령어의 연산 종료를 계속 기다릴 수 있다. 이 때, 해당 오퍼랜드 컬렉터는 이미 다음 명령어에 대해서 할당되었기 때문에 모든 소스 오퍼랜드가 준비된 다른 명령어에 대해서는 할당될 수 없다. 따라서, 모든 소스 오퍼랜드가 준비된 다른 명령어의 실행이 늦춰질 수 있다.

[0068] 동작 ④에서, 스코어보드(420)를 이용하면 긴 대기 시간이 발생하는 데이터 의존성을 갖는 명령어가 확인될 수 있다. 확인된 결과 다음 명령어가 긴 대기 시간이 발생하는 데이터 의존성을 갖지 않는 경우에만 오퍼랜드 컬렉터가 다음 명령어에 대해서 미리 할당될 수 있다. 다시 말해, 다음 명령어에 대하여 데이터 의존성이 발생하였고, 이전 명령어의 연산 시간이 임계값을 초과하는 경우, 오퍼랜드 컬렉터가 다음 명령어에 대해 할당되지 않을 수 있다.

[0069] 일 실시예에 따르면, 모든 명령어들이 긴 대기 시간이 발생하는 데이터 의존성을 갖는 경우, 오퍼랜드 컬렉터들은 명령어들의 순차적인 실행 순서에 따라서 미리 할당될 수 있다.

[0070] 동작 ④를 통해서, 긴 대기 시간을 갖는 다음 명령어에 대해서 오퍼랜드 컬렉터를 미리 할당하지 않음으로써 모든 소스 오퍼랜드들이 준비된 다른 명령어에 대해서 오퍼랜드 컬렉터가 할당될 수 없는 문제를 피할 수 있다.

[0071] 이하에서는, 이전 명령어의 연산 결과를 포워딩하는 제1 로직 회로(450)에 대해서 더 설명하도록 하겠다.

[0073] 도 5는 본 개시의 일 실시예에 따른 제1 로직 회로를 도시한 도면이다.

[0074] 도 5를 참조하면, 제1 로직 회로(500)가 도시된다.

[0075] 제1 로직 회로(500)는 오퍼랜드 컬렉터들의 개수에 대응하는 엔트리들을 가진 CAM(content addressable memory)(501)을 포함할 수 있다. 제1 로직 회로(500)는 복수의 논리 게이트들을 포함할 수 있다. 워프 스케줄러(520)가 오퍼랜드 컬렉터를 미리 할당할 때 엔트리에 포워딩 데이터를 저장할 수 있다. 포워딩 데이터는 복수의 소스 오퍼랜드들 중 하나 이상을 제외한 나머지 소스 오퍼랜드(즉, 레지스터 파일에 아직 준비되지 않은 소스 오퍼랜드)의 식별자 데이터(이하, TAG), 오퍼랜드 컬렉터(즉, 미리 할당된 오퍼랜드 컬렉터)의 식별자 데이터(이하, OCID(operand collector identifier)) 및 연산 결과를 레지스터 파일에 저장할 것 인지에 대한 데이터(이하, DR bit)를 포함할 수 있다.

[0076] 제1 로직 회로(500)는 이전 명령어의 연산 결과에 대한 목적지를 결정할 수 있다. 제1 로직 회로(500)는 엔트리에 저장된 포워딩 데이터를 이용하여 이전 명령어의 연산 결과에 대한 목적지를 결정할 수 있다.

[0077] 이하에서는 제1 로직 회로(500)의 동작 방법을 동작 ① 내지 동작 ⑥를 이용하여 설명하도록 하겠다. 다만, 동작 ① 내지 동작 ⑥는 순차적으로 수행될 수도 있으나, 반드시 순차적으로 수행되는 것은 아니다. 예를 들어, 각 동작들의 순서가 변경될 수도 있으며, 적어도 두 동작들이 병렬적으로 수행될 수도 있다.

[0078] 오퍼랜드 컬렉터(530)를 미리 할당할 때, 워프 스케줄러(520)는 할당된 오퍼랜드 컬렉터(530)에 대응하는 Warp ID(예: 6 bits) 및 아직 준비되지 않은 소스 오퍼랜드의 Reg ID(예: 8 bits), OCID 및 DR bit를 제1 로직 회로(500)에 전송할 수 있다.

[0079] 동작 ①에서, 엔트리는 OCID를 이용하여 인덱싱될 수 있다. 동작 ②에서, Warp ID 및 Reg ID 가 연결되어 엔트리에 대한 TAG로 저장될 수 있다. 동작 ③에서 OCID 및 DR bit도 TAG와 동일한 엔트리에 저장될 수 있다.

[0080] 이전 명령어의 연산이 종료하면, 제1 로직 회로(500)는 실행 유닛(510)으로부터 Warp ID 및 Reg ID와 함께 연산 결과(이하, DATA)를 수신할 수 있다. 동작 ④에서, DATA가 전송될 오퍼랜드 컬렉터(530)를 결정하기 위해서 제1 로직 회로(500)는 포워딩 데이터와 실행 유닛(510)으로부터 수신한 연산 결과에 대한 식별자 데이터를 비교할 수 있다. DATA가 전송될 오퍼랜드 컬렉터(530)를 결정하기 위해서 제1 로직 회로(500)는 각 엔트리와 실행 유

닛(510)으로부터 수신한 Warp ID 및 Reg ID를 이용하여 매칭되는 TAG를 결정할 수 있다. 제1 로직 회로(500)는 수신한 Warp ID 및 Reg ID를 연결하고, AND 게이트를 이용하여 연결된 Warp ID 및 Reg ID를 각 엔트리의 TAG와 비교하여 매칭되는 TAG를 결정할 수 있다. 동작 ⑤에서, 비교 결과가 참(true)인 엔트리에 대응하는 오퍼랜드 컬렉터(530)에 OCID를 이용하여 DATA가 전달될 수 있다. 비교 결과가 참이 아닌 경우(즉, 거짓인 경우) DATA는 오퍼랜드 컬렉터(530)에 전달되지 않을 수 있다.

[0081] 제1 로직 회로(500)는 포워딩 데이터에 기반하여 DATA를 레지스터 파일에 저장할지 여부를 판단할 수 있다. 제1 로직 회로(500)는 DR bit를 이용하여 DATA를 레지스터 파일에 저장할지 여부를 판단할 수 있다. 기본적으로, DATA는 항상 레지스터 파일에 쓰여지는 것으로 설정될 수 있다. 그러나, 이후의 명령어에서 DATA를 참조하지 않는 경우 또는 DATA가 다음 명령어의 연산 결과에 의해 바로 업데이트 되는 경우, DATA는 레지스터 파일에 저장될 필요가 없을 수 있다. 이를 위해 DR bit가 사용될 수 있다. 동작 ⑥에서, AND 게이트를 이용하여 동작 ④의 비교 결과와 DR bit를 비교할 수 있다. 비교 결과가 참인 경우 DATA는 레지스터 파일에 저장되지 않을 수 있다. 비교 결과가 참이 아닌 경우(false) DATA는 뱅크 아비트레이터(bank arbitrator)를 통해 레지스터 파일에 저장될 수 있다.

[0082] 이하에서는 DR bit를 결정하는 제2 로직 회로에 대해서 설명하도록 하겠다.

[0084] 도 6은 본 개시의 일 실시예에 따른 제2 로직 회로를 도시한 도면이다.

[0085] 이하에서는 제2 로직 회로(600)의 동작 방법을 동작 ⑦ 내지 동작 ⑧를 이용하여 설명하도록 하겠다. 다만, 동작 ⑦ 내지 동작 ⑧는 순차적으로 수행될 수도 있으나, 반드시 순차적으로 수행되는 것은 아니다. 예를 들어, 각 동작들의 순서가 변경될 수도 있으며, 적어도 두 동작들이 병렬적으로 수행될 수도 있다.

[0086] 제2 로직 회로(600)는 작은 크기의 SRAM(static random access memory)(620)을 포함할 수 있다. SRAM(620)은 복수의 엔트리들을 포함할 수 있다. 각 엔트리는 복수의 비트들을 포함할 수 있다. 복수의 비트들의 개수는 명령어가 필요로 하는 소스 오퍼랜드의 개수에 대응할 수 있다. 예를 들어, 명령어가 필요로 하는 소스 오퍼랜드의 개수가 3개인 경우 복수의 비트들의 개수는 3개일 수 있다. 복수의 비트들은 대응하는 복수의 소스 오퍼랜드들에 대한 DR bit일 수 있다.

[0087] 복수의 비트들 각각은 대응하는 소스 오퍼랜드를 레지스터 파일에 저장할 필요가 있는지 여부를 나타낼 수 있다. 복수의 비트들 각각은 대응하는 소스 오퍼랜드가 한 번만 사용되는지 여부를 나타낼 수 있다. 복수의 비트들 각각은 대응하는 소스 오퍼랜드가 데이터 의존성이 있는지 여부를 나타낼 수 있다. 소스 오퍼랜드를 레지스터 파일에 저장할 필요가 있는지 여부는 컴파일 타임에서 탐색될 수 있다.

[0088] 초기 커널(kernel) 생성 과정에서 DR bit 벡터가 제2 로직 회로(600)의 SRAM(620)에 로딩될 수 있다. 워프 스케줄러(610)는 오퍼랜드 컬렉터가 미리 할당되는 다음 명령어가 확인되면, 레지스터 파일에 저장되지 않은 소스 오퍼랜드의 위치와 함께 해당 명령어에 대한 프로그램 카운터(program counter, PC)를 제2 로직 회로(600)에 전송할 수 있다.

[0089] 동작 ⑦에서, SRAM(620)에 포함되는 엔트리들의 인덱스는 프로그램 카운터를 임의의 비트만큼 이동(shift)하여 계산될 수 있다. 임의의 비트는 명령어의 비트수에 기반하여 결정될 수 있다. 예를 들어, 64 비트의 명령어에서 임의의 비트는 3 비트로 결정될 수 있다.

[0090] 엔트리에 포함되는 3개의 DR bit는 MUX(multiplexer)의 입력 버퍼로 래치(latch)될 수 있다. 동작 ⑧에서, 워프 스케줄러(610)로부터 MUX로 전송된 레지스터 파일에 저장되지 않은 소스 오퍼랜드의 위치(즉, 아직 준비되지 않은 소스 오퍼랜드의 위치)(도 6의 Src Idx)가 MUX에 대한 입력들 중 어느 하나를 선택하는데 이용될 수 있다.

[0091] MUX에서 선택된 DR bit는 워프 스케줄러(610)로 전송될 수 있다. 워프 스케줄러(610)는 선택된 DR bit를 제1 로직 회로로 전달할 수 있다(도 5의 동작 ③). DR bit는 제1 로직 회로에서 이전 명령어의 연산 결과를 레지스터 파일에 전송할지 여부를 판단하는 데 이용될 수 있다.

[0093] 도 7은 본 개시의 일 실시예에 따른 전자 장치의 동작 방법을 설명하기 위한 플로우차트이다.

[0094] 이하 실시예에서 각 동작들은 순차적으로 수행될 수도 있으나, 반드시 순차적으로 수행되는 것은 아니다. 예를 들어, 각 동작들의 순서가 변경될 수도 있으며, 적어도 두 동작들이 병렬적으로 수행될 수도 있다.

[0095] 단계(710)에서, 워프 스케줄러가 스코어보드를 이용하여 다음 명령어에 대한 복수의 소스 오퍼랜드들 중 하나 이상이 레지스터 파일에 저장되어 있는지 여부를 판단할 수 있다.

[0096] 단계(720)에서, 복수의 소스 오퍼랜드들 중 하나 이상이 레지스터 파일에 저장되어 있는 경우, 다음 명령어에 대하여 오퍼랜드 컬렉터를 할당할 수 있다.

[0097] 단계(730)에서, 이전 명령어의 연산 결과를 오퍼랜드 컬렉터에 전송하는 제1 로직 회로를 이용하여 연산 결과를 오퍼랜드 컬렉터에 전송할 수 있다.

[0098] 단계(740)에서, 오퍼랜드 컬렉터에 로딩된 연산 결과 및 복수의 소스 오퍼랜드들 중 하나 이상을 이용하여 다음 명령어에 대한 연산을 수행할 수 있다.

[0100] 한편, 본 발명에 따른 방법은 컴퓨터에서 실행될 수 있는 프로그램으로 작성되어 마그네틱 저장매체, 광학적 판독매체, 디지털 저장매체 등 다양한 기록 매체로도 구현될 수 있다.

[0101] 본 명세서에 설명된 각종 기술들의 구현들은 디지털 전자 회로조직으로, 또는 컴퓨터 하드웨어, 펌웨어, 소프트웨어로, 또는 그들의 조합들로 구현될 수 있다. 구현들은 데이터 처리 장치, 예를 들어 프로그램가능 프로세서, 컴퓨터, 또는 다수의 컴퓨터들의 동작에 의한 처리를 위해, 또는 이 동작을 제어하기 위해, 컴퓨터 프로그램 제품, 즉 정보 캐리어, 예를 들어 기계 판독가능 저장 장치(컴퓨터 판독가능 매체) 또는 전파 신호에서 유형적으로 구체화된 컴퓨터 프로그램으로서 구현될 수 있다. 상술한 컴퓨터 프로그램(들)과 같은 컴퓨터 프로그램은 컴파일된 또는 인터프리트된 언어들을 포함하는 임의의 형태의 프로그래밍 언어로 기록될 수 있고, 독립형 프로그램으로서 또는 모듈, 구성요소, 서브루틴, 또는 컴퓨팅 환경에서의 사용에 적절한 다른 유닛으로서 포함하는 임의의 형태로 전개될 수 있다. 컴퓨터 프로그램은 하나의 사이트에서 하나의 컴퓨터 또는 다수의 컴퓨터들 상에서 처리되도록 또는 다수의 사이트들에 걸쳐 분배되고 통신 네트워크에 의해 상호 연결되도록 전개될 수 있다.

[0102] 컴퓨터 프로그램의 처리에 적절한 프로세서들은 예로서, 범용 및 특수 목적 마이크로프로세서들 둘 다, 및 임의의 종류의 디지털 컴퓨터의 임의의 하나 이상의 프로세서들을 포함한다. 일반적으로, 프로세서는 판독 전용 메모리 또는 랜덤 액세스 메모리 또는 둘 다로부터 명령어들 및 데이터를 수신할 것이다. 컴퓨터의 요소들은 명령어들을 실행하는 적어도 하나의 프로세서 및 명령어들 및 데이터를 저장하는 하나 이상의 메모리 장치들을 포함할 수 있다. 일반적으로, 컴퓨터는 데이터를 저장하는 하나 이상의 대량 저장 장치들, 예를 들어 자기, 자기-광 디스크들, 또는 광 디스크들을 포함할 수 있거나, 이것들로부터 데이터를 수신하거나 이것들에 데이터를 송신하거나 또는 양쪽으로 되도록 결합될 수도 있다. 컴퓨터 프로그램 명령어들 및 데이터를 구체화하는데 적절한 정보 캐리어들은 예로서 반도체 메모리 장치들, 예를 들어, 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(Magnetic Media), CD-ROM(Compact Disk Read Only Memory), DVD(Digital Video Disk)와 같은 광 기록 매체(Optical Media), 플롭티컬 디스크(Floptical Disk)와 같은 자기-광 매체(Magneto-Optical Media), 롬(ROM, Read Only Memory), 램(RAM, Random Access Memory), 플래시 메모리, EPROM(Erasable Programmable ROM), EEPROM(Electrically Erasable Programmable ROM) 등을 포함한다. 프로세서 및 메모리는 특수 목적 논리 회로조직에 의해 보충되거나, 이에 포함될 수 있다.

[0103] 또한, 컴퓨터 판독가능 매체는 컴퓨터에 의해 액세스될 수 있는 임의의 가용매체일 수 있고, 컴퓨터 저장매체 및 전송매체를 모두 포함할 수 있다.

[0104] 본 명세서는 다수의 특정한 구현물의 세부사항들을 포함하지만, 이들은 어떠한 발명이나 청구 가능한 것의 범위에 대해서도 제한적인 것으로서 이해되어서는 안되며, 오히려 특정한 발명의 특정한 실시형태에 특유할 수 있는 특징들에 대한 설명으로서 이해되어야 한다. 개별적인 실시형태의 문맥에서 본 명세서에 기술된 특정한 특징들은 단일 실시형태에서 조합하여 구현될 수도 있다. 반대로, 단일 실시형태의 문맥에서 기술한 다양한 특징들 역시 개별적으로 혹은 어떠한 적절한 하위 조합으로도 복수의 실시형태에서 구현 가능하다. 나아가, 특징들이 특정한 조합으로 동작하고 초기에 그와 같이 청구된 바와 같이 묘사될 수 있지만, 청구된 조합으로부터의 하나 이상의 특징들은 일부 경우에 그 조합으로부터 배제될 수 있으며, 그 청구된 조합은 하위 조합이나 하위 조합의 변형물로 변경될 수 있다.

[0105] 마찬가지로, 특정한 순서로 도면에서 동작들을 묘사하고 있지만, 이는 바람직한 결과를 얻기 위하여 도시된 그 특정한 순서나 순차적인 순서대로 그러한 동작들을 수행하여야 한다거나 모든 도시된 동작들이 수행되어야 하는 것으로 이해되어서는 안 된다. 특정한 경우, 멀티태스킹과 병렬 프로세싱이 유리할 수 있다. 또한, 상술한 실시형태의 다양한 장치 컴포넌트의 분리는 그러한 분리를 모든 실시형태에서 요구하는 것으로 이해되어서는 안되며, 설명한 프로그램 컴포넌트와 장치들은 일반적으로 단일의 소프트웨어 제품으로 함께 통합되거나 다중 소프트웨어 제품에 패키징 될 수 있다는 점을 이해하여야 한다.

[0106] 한편, 본 명세서와 도면에 개시된 본 발명의 실시 예들은 이해를 돕기 위해 특정 예를 제시한 것에 지나지 않으

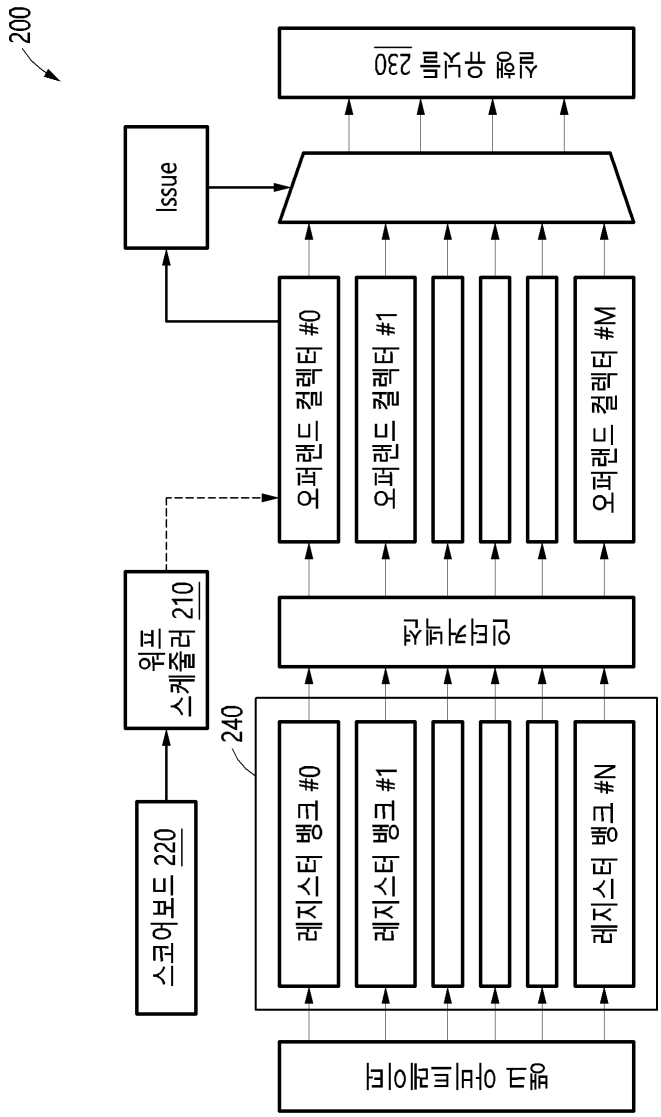
며, 본 발명의 범위를 한정하고자 하는 것은 아니다. 여기에 개시된 실시 예들 이외에도 본 발명의 기술적 사상에 바탕을 둔 다른 변형 예들이 실시 가능하다는 것은, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에게 자명한 것이다.

도면

도면1



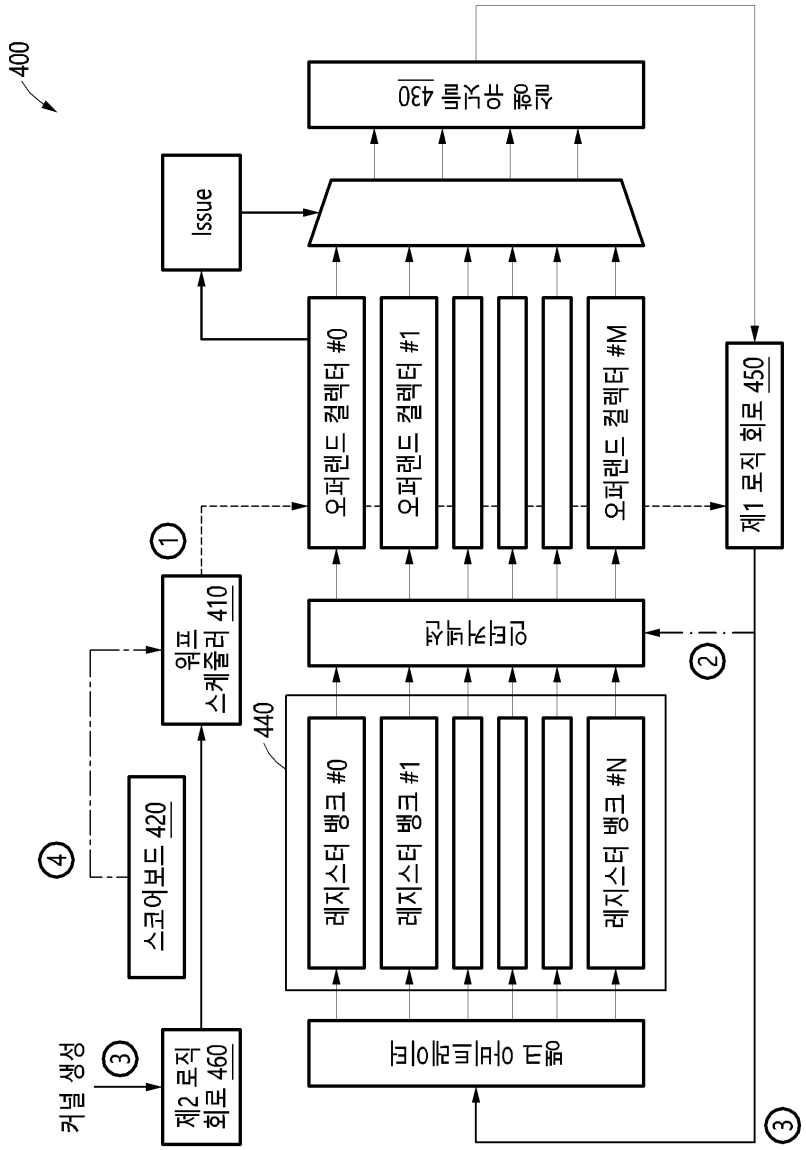
도면2



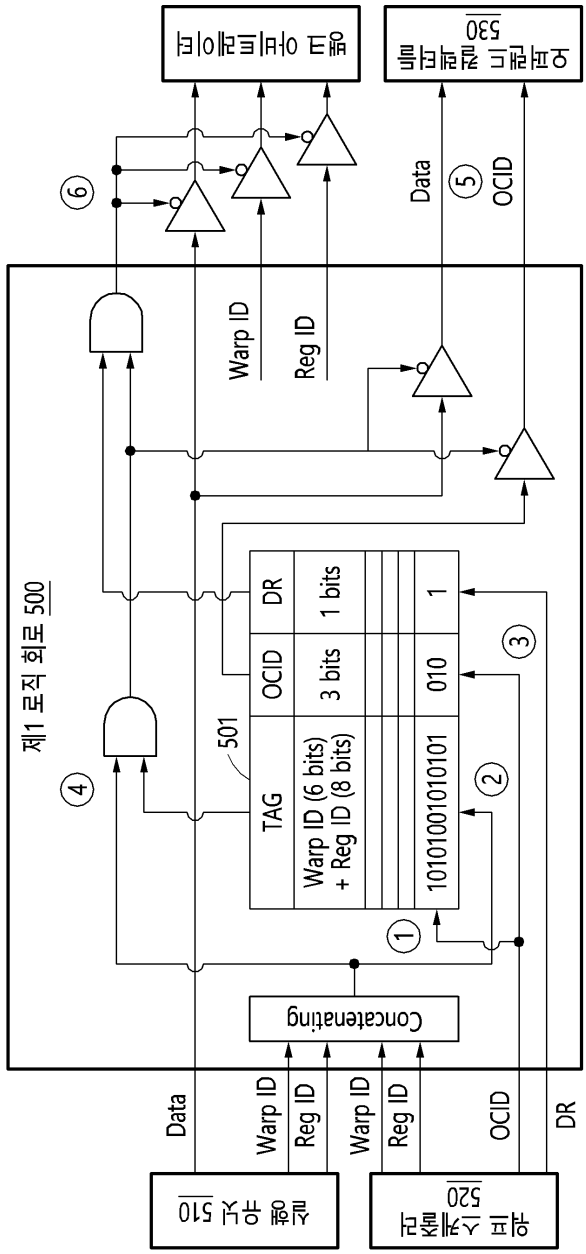
도면3

명령어 [OP, DST, SRC1, SRC2]	
i1	ld, global R5 [0x030]
i2	ld, global R8 [0x034]
i3	sub **R9**, R5, R8 ①
i4	sub *R7*, R5, **R9** ②
i5	sub *R7*, R5, *R7*

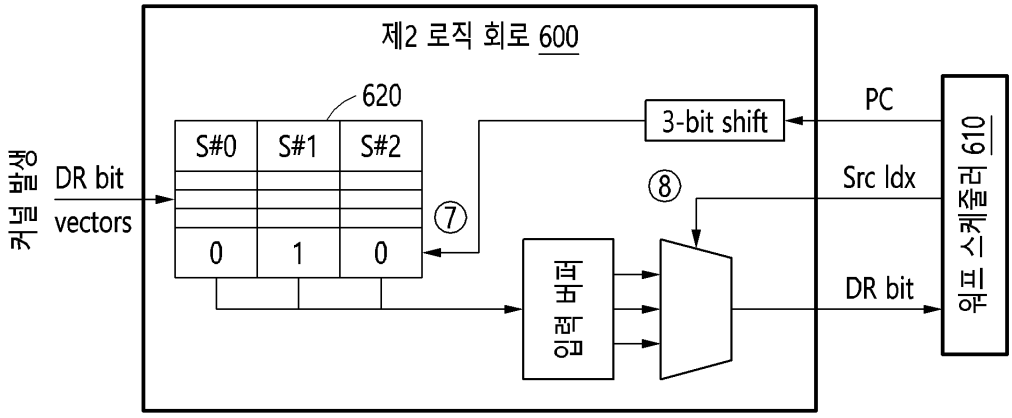
도면4


400
실행 유닛 430
Issue
오퍼랜드 컬렉터 #0
오퍼랜드 컬렉터 #1
오퍼랜드 컬렉터 #M
제1 로직 회로 450
①
워프 스케줄러 410
인터커넥트
②
440
레지스터 뱅크 #0
레지스터 뱅크 #1
레지스터 뱅크 #N
④
스코어보드 420
메모리 유닛
③
커널 생성
③
제2 로직 회로 460

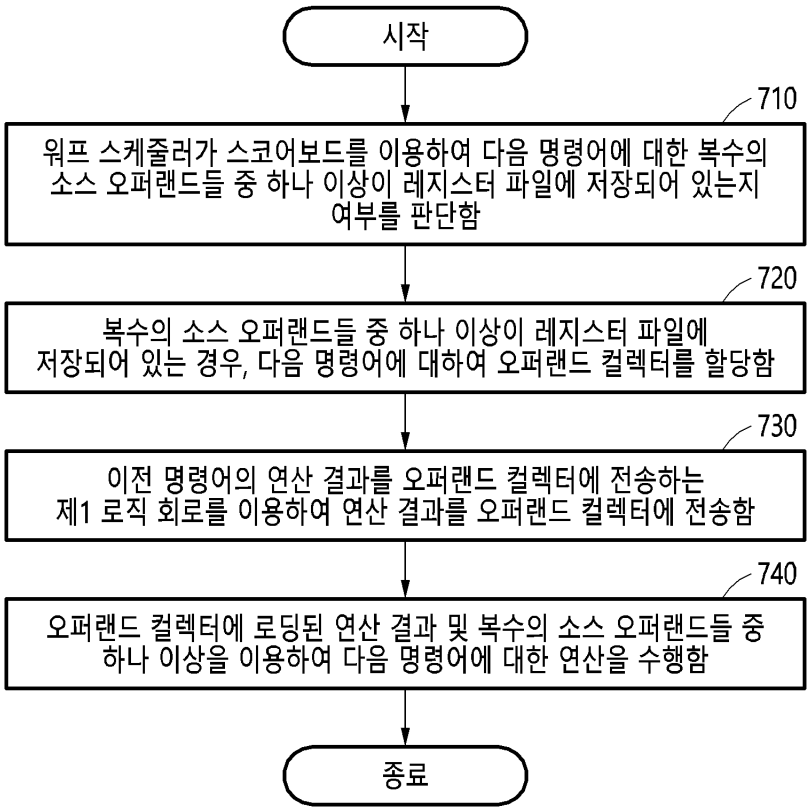
도면5


제1 로직 회로 500
501
TAG
Warp ID (6 bits) + Reg ID (8 bits)
OCID
3 bits
DR
1 bits
101010010101010101
010
1
Concatenating
Warp ID
Reg ID
Data
OCID
DR
Warp ID
Reg ID
Data
OCID
① ② ③ ④ ⑤ ⑥

도면6


제2 로직 회로 600
620
S#0
S#1
S#2
0
1
0
커널 발생
DR bit
vectors
⑦
3-bit shift
PC
⑧
Src Idx
DR bit
맵핑
워프 스케줄러 610

도면7


시작
710
워프 스케줄러가 스코어보드를 이용하여 다음 명령어에 대한 복수의 소스 오퍼랜드들 중 하나 이상이 레지스터 파일에 저장되어 있는지 여부를 판단함
720
복수의 소스 오퍼랜드들 중 하나 이상이 레지스터 파일에 저장되어 있는 경우, 다음 명령어에 대하여 오퍼랜드 컬렉터를 할당함
730
이전 명령어의 연산 결과를 오퍼랜드 컬렉터에 전송하는 제1 로직 회로를 이용하여 연산 결과를 오퍼랜드 컬렉터에 전송함
740
오퍼랜드 컬렉터에 로딩된 연산 결과 및 복수의 소스 오퍼랜드들 중 하나 이상을 이용하여 다음 명령어에 대한 연산을 수행함
종료