**Week 6 Assignment**

**1. 컴퓨터 시스템 구조의 분류 방법인 Flynn's Taxonomy (Flynn의 분류 방법)의 분류 기준과 제시하는 4가 지 시스템 구조, 각 분류가 가지는 의미를 서술하세요.**

(답안)

Flynn은 컴퓨터 시스템을 분류하기 위한 기준으로 Instruction Stream와 Data Stream을 제안함. 여기서 Instruction 이라 함은 명령어를 의미하고 Data라 함은 Instruction의 Operand들을 의미함 또한 Stream은 컴퓨터에 의해 처 리되는 연속된 Instruction 혹은 Data의 흐름을 의미함.

Instruction Stream과 Data Stream은 각각 Stream 개수에 따라 Single 혹은 Multiple로 구분됨. Instruction Stream 과 Data Stream의 조합에 따라 4 종류로 분류되며 각 분류와 의미는 아래와 같음.

- Single Instruction stream, Single Data stream **(SISD)**

한 번에 하나의 Instruction만이 수행되며 하나의 Data Stream을 처리함. SISD에서는 명령어들이 순차 처리되며 병렬성이 없음. 구식의 단일 CPU (Single Core Processor)들이 이에 속함.

- Single Instruction stream, Multiple Data stream **(SIMD)**

하나의 Instruction Stream이 실행되지만 다수의 Data Stream을 처리함. 같은 Instruction이 동시에 여 러 Data에 대해 수행되며 Cray-1와 같은 Vector Processor, GPU 등이 이에 속함.

- Multiple Instruction stream, Single Data stream **(MISD)**

다수의 Instruction들이 동시에 하나의 Data Stream를 처리함. 서로 다른 Instruction들이 같은 Data에 대해 수행됨. 특수목적 혹은 실험적 설계 사례 등이 존재하지만 널리 사용되지 않음.

- Multiple Instruction stream, Multiple Data stream **(MIMD)**

다수의 Instruction Stream이 동시에 실행될 수 있고 다수의 Data Stream에 대해 처리함. 다수의 프로 세서들이 서로 다른 Instruction들을 서로 다른 Data에 대해 수행할 수 있음. 대부분의 Supercomputer들이 컴퓨터들을 네트워크로 묶어 구현하고 있음.

**2. FI(Fetch Instruction), DA(Decode and calculate effective Address), FO(Fetch Operand), EX(EXecute instruction) 총 4개 세그먼트로 이루어진 파이프라인이 있습니다. (4.7 명령어 파이프라인 슬라이드 참조) Branch 명령어가 포함되지 않은 7개의 Instruction을 처리하는 과정을 표로 나타내세요.**

(답안)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Step | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Instruction 1 2  3  4  5  6  7 | FI | DA | FO | EX |  |  |  |  |  |  |
|  | FI | DA | FO | EX |  |  |  |  |  |
|  |  | FI | DA | FO | EX |  |  |  |  |
|  |  |  | FI | DA | FO | EX |  |  |  |
|  |  |  |  | FI | DA | FO | EX |  |  |
|  |  |  |  |  | FI | DA | FO | EX |  |
|  |  |  |  |  |  | FI | DA | FO | EX |

**3. 2번 문제와 같은 파이프라인 구조에 분기 예측을 추가하여 FI 세그먼트에서 분기로 인해 실행할 Instruction을 예측할 수 있게 되었습니다. 분기 예측이 추가된 파이프라인에서는 예측한 Instruction에 대한 DA, FO를 Branch Instruction의 EX 전에 미리 수행합니다. 또한 Branch Instruction의 EX 후에 분 기 예측이 실패할 경우, 먼저 실행해둔 DA와 FO를 폐기하고 다음 Step에서 새로운 Instruction에 대해 FI 부터 다시 수행해야 합니다. (Step 5에서 분기 명령어에 대해 EX를 수행해 분기 예측이 틀림을 알았다 면 Step 6에서 새로운 명령어에 대해 FI를 수행해야함.) 이때 3번째 Instruction이 분기 명령어인 총 7개 의 Instruction을 처리하는 과정을 표로 나타내세요.**

(답안)

Branch Prediction이 추가됨으로써 예측이 성공했을 경우에는 문제 2에 제시한 표와 같이 실행됨.

그러나 예측에 실패했을 경우 아래에 제시된 표와 같이 DA, FO를 폐기하고 새로운 Instruction을 실행하므로 Step이 추가됨.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Step | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| Instruction 1 2  **(Branch Inst.)** 3 4  5  6  7 | FI | DA | FO | EX |  |  |  |  |  |  |  |  |  |
|  | FI | DA | FO | EX |  |  |  |  |  |  |  |  |
|  |  | FI | DA | FO | EX |  |  |  |  |  |  |  |
|  |  |  | ~~FI~~ | ~~DA~~ | ~~FO~~ | FI | DA | FO | EX |  |  |  |
|  |  |  |  | ~~FI~~ | ~~DA~~ |  | FI | DA | FO | EX |  |  |
|  |  |  |  |  | ~~FI~~ |  |  | FI | DA | FO | EX |  |
|  |  |  |  |  |  |  |  |  | FI | DA | FO | EX |

\* Branch Prediction 실패로 폐기된 작업은 취소선으로 표시.

\*\* Step 7부터 실행되는 Instruction 4, 5, 6, 7은 Step 4, 5, 6에서 Fetch된 Instruction과 달리 예측하지 못한 분기 쪽 의 Instruction임.

**4. 캐시 하나(L1)와 메인 메모리(M) 하나를 가진 CPU가 있습니다. 캐시의 Hit Ratio가 0.5 이고 캐시에 접 근하는데 걸리는 시간은 10 ns, 메인 메모리에 접근하는 시간은 800 ns 입니다. 이 때 평균 데이터 접근 시간을 구하세요. (캐시에 데이터가 존재하는지 여부는 캐시에 우선 접근하고 나서 알게 됩니다.)** (답안)

데이터의 접근이 캐시 접근, 데이터가 없으면 메인 메모리에 접근 순으로 이루어지므로 평균 데이터 접근 시간은 ������ℎ���������������������� + (1 − ����������������) × ��������������������������������으로 구할 수 있음.

따라서 10 + (1 − 0.5) × 800 = 410 ���� 임.

**5. 4번 문제에 제시된 시스템에서 캐시와 메인 메모리 사이에 캐시를 두개 더 추가했습니다. 따라서 시스 템에는 캐시 L1, 캐시 L2, 캐시 L3, 메인 메모리 M이 있습니다. L1의 접근 시간은 10 ns, L2의 접근시간 은 50ns, L3의 접근 시간은 100ns, M의 접근 시간은 800 ns입니다. 그리고 L1의 Hit Ratio는 0.5, L2의 Hit Ratio는 0.6, L3의 Hit Ratio는 0.8 입니다. 이때 평균 데이터 접근 시간을 구하세요. (접근 순서는 L1**

**→ L2 → L3 → M입니다.)**

(답안)

데이터 접근은

(1) L1 접근

(2) L1에서 발견하지 못하면 L2 접근

(3) L2에서 발견하지 못하면 L3 접근

(4) L3에서 발견하지 못하면 메인 메모리 접근

순으로 이루어짐.

평균 데이터 접근 시간은 아래와 같이 정리할 수 있음. (L1, L2, L3캐시의 접근 시간은 ��1������������, ��2������������, ��3������������ 로 표시하고 각 캐시의 Hit Ratio는 ��1����������������, ��2����������������, ��3����������������로 표시함. 메인 메모리 접근 시간은 ��������������로 표 시함.)

��1������������ + (1 − ��1����������������)(��2������������ + (1 − ��2����������������)(��3������������ + (1 − ��3����������������)��������������))

따라서 10 + (1 − 0.5)(50 + (1 − 0.6)(100 + (1 − 0.8)800)) = 10 + 0.5(50 + 0.4(100 + 0.2 × 800)) = 87���� 임.

**6. 3개의 멀티프로세서 시스템이 있습니다. 첫 번째 시스템(S1)은 4개의 프로세서를 갖고있으며 두 번째 시 스템(S2)은 8개, 마지막 시스템(S3)은 16개를 가지고 있습니다. 세 시스템 모두 하이퍼큐브 연결 방식을 사용하고 있습니다. 시스템 S1, S2, S3에 대해 두 프로세서 간의 (1) 최단거리, (2) 최장거리 그리고 (3) 임의의 두 프로세서 간의 거리의 평균을 표로 정리하세요. (거리는 프로세서를 점, 연결을 선으로 나타 냈을 때, 한 점에서 다른 점으로 이동하기 위해 거치는 변의 개수로 합니다. 임의의 두 프로세서 간의 거리의 평균은 가능한 모든 두 프로세서의 조합에 대해 구한 거리들의 평균입니다.)**

(답안)

|  |  |  |  |
| --- | --- | --- | --- |
|  | S1 | S2 | S3 |
| (1) 최단거리 | 1 | 1 | 1 |
| (2) 최장거리 | 2 | 3 | 4 |
| (3) 거리의 평균 | 8  6≈ 1.3333 … | 48  28 ≈ 1.71428 … | 256  120 ≈ 2.133333 …. |

각 시스템 S1, S2, S3들은 프로세서들을 꼭짓점으로 하여 Hypercube 모양으로 상호 연결됨. 따라서 S1은 정사각형, S2 정육면체, S3는 정팔포체 모양이 됨.

**1. 각 시스템의 연결 구조(정사각형, 정육면체, 정팔포체)의 투영도로부터 직접 계산할 수 있음.** 최단거리와 최장거리는 투영도에서 두 정점(프로세스) 간에 가능한 가장 짧은 경로와 가장 긴 경로를 찾아 해결 할 수 있음.

거리의 평균은 거리별로 가능한 프로세서의 조합을 통해 구할 수 있음.

**2. 발생할 수 있는 거리 별로 나누어 정사각형 (시스템 S1)에서부터 정육면체 (S2), 정팔면체 (S3)에 대해 계산할 수 있음.**

- 정사각형에서 나타나는 최단 거리는 1, 최장 거리는 2로 쉽게 구할 수 있음.

S1에서 거리가 1인 두 프로세서의 조합의 개수를 C[S1][1], 거리가 2인 두 프로세서의 조합 개수를 C[S1][2]라고 하면,

C[S1][1] = 4, C[S1][2] = 2임.

- 정육면체는 윗면과 아랫면에 정사각형을 배치하고 각 정점을 잇는 간선을 추가한 것과 같음. 그렇다면 정육면체(S2)에 대해 C[S2][1]는 두 개의 정사각형 (2 x C[S1][1])과 각 꼭지점을 잇는 간선 (C[S1][1])으로 C[S2][1] = 3 x C[S1][1]임.

정육면체에서 C[S2][2], 거리가 2인 두 프로세서 간의 이동 경로는 총 두가지 경우가 존재할 수 있음. (1) 윗면(혹은 아랫면) 정사각형에서 2만큼 이동함.

(2) 윗면(혹은 아랫면) 정사각형에서 1만큼 이동한 후 아랫면(혹은 윗면)으로 1 이동함 즉, C[S2][2]는 윗면/아랫면 정사각형에서 거리 2를 갖는 조합(2 x C[S1][2])과 윗면/아랫면 정사각형에서 1만큼 움 직이고 반대편 정사각형으로 움직이는 조합(2 x C[S1][1])의 합으로 나타낼 수 있음.

따라서 C[S2][2] = 2 x C[S1][1] + 2 x C[S1][2]임.

앞서 언급한 (1)경우는 아직 반대편 정사각형으로(윗면->아랫면, 아랫면->윗면) 넘어가지 않았기 때문에 1회 더 움직여 거리 3의 조합을 만들 수 있음.

따라서 C[S2][3] = 2 x C[S1][2] 임.

- 정팔포체 또한 정육면체와 같은 방법으로 정육면체를 두개 배치하고 각 정점을 잇는 간선을 추가한 것 과 같음.

정리하면,

C[S3][1] = 3 x C[S2][1] (정육면체 1 + 정육면체 2 + 두 정육면체 사이의 간선)

C[S3][2] = 2 x C[S2][1] + 2 x C[S2][2] (한쪽에서 1만큼 이동하고 반대편으로 1 이동 + 한쪽에서 2만큼 이동) C[S3][3] = 2 x C[S2][2] + 2 x C[S2][3] (한쪽에서 2만큼 이동하고 반대편으로 1 이동 + 한쪽에서 3만큼 이동) C[S3][4] = 2 x C[S2][3] (한쪽에서 3만큼 이동하고 반대편으로 1 이동)임.

위 계산과정들 중 한쪽에서 1, 건너가는데 1, 반대편에서 1만큼 이동하는 경우를 고려하지 않는 이유는 두 정점 을 정하면 거리는 어떤 경로를 타던 같고,

(1) 한쪽에서 1, 건너가는데 1, 반대편에서 1

(2) 한쪽에서 2, 건너가는데 1

(3) 건너가는데 1, 반대편에서 2

세가지 중 어떤 것이 되든 마지막에 건너는 방식으로 바꾸어도 문제가 없기 때문임.

**3. Hamming Distance를 이용하여 계산할 수 있음.**

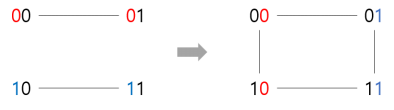
Hamming Distance을 간략히 말하면, 길이가 같은 두 문자열에서 같은 위치에 서로 다른 문자가 있는 개수임. 예를 들면, 11001과 10111의 Hamming Distance는 3임. 두번째부터 네번째 문자가 서로 다르기 때문.

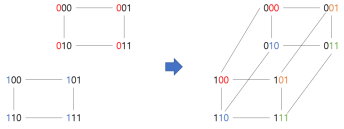
이를 이용하여, 두개 프로세서가 있는 경우부터 시작함.

각 프로세서를 정점으로 삼아, 0과 1을 이름으로 붙이고 둘 사이에 간선을 추가함.



앞서 제시한 풀이처럼 선분을 두개 배치하고 한쪽에는 0을, 한쪽에는 1을 맨 앞에 추가하고 앞에 추가한 0/1을 제외한 나머지 부분이 같은 정점들 사이에 간선을 추가함.

이를 정점의 개수가 원하는 만큼 늘어날 때까지 정사각형, 정육면체에 대해 반복함.

…

이런 방식으로 간선과 정점을 추가해 나가면, 거리가 1인 것은 두 정점에 붙은 이름에 Hamming Distance가 1이 됨. (항상 추가되는 간선은 맨 앞에 새로 붙인 0/1을 제외하고 나머지 부분이 같아야 하기 때문) 또한, 거리가 2인 두 정점은 Hamming Distance가 2임.

각 정점에 2진법으로 번호를 부여하면 앞서 언급한 특성을 이용해 임의의 두 정점 간에 거리의 평균을 구할 수 있음.

예시로 S3, 정점이 16개인 경우, 0~15를 사용해 번호를 부여하고(4bit), 모든 조합에 대해 Hamming Distance의 합을 가능한 조합의 개수로 나누어 구할 수 있음.