

위치	오류유형	수정 전	수정 후																																																				
71p	문제-본문	<p style="text-align: center;">독학사 컴퓨터공학과 4단계 통합컴퓨터시스템</p> <p>연산 33의 보수로 표현한 뒤 덧셈연산을 수행하면 된다. 올림수인 캐리가 발생하지만 캐리에 대한 처리 없이 올바른 연산결과가 나온다.</p> <p>반대로 음수 + 양수이고 음수의 크기가 더 큰 경우인 -33 + 25을 살펴보자.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>2진수</th> <th>10진수</th> </tr> </thead> <tbody> <tr> <td>MSB 1 101111 + 0 0011001 ----- 1 1111000</td> <td>-33 + 25 ----- - 8</td> </tr> </tbody> </table> <p>앞의 연산과 동일하게 33을 2의 보수로 표현한 뒤 덧셈연산을 수행한다. 맨 앞의 MSB가 1이므로 음수를 의미한다. 음수는 2의 보수로 표현되어 있으므로 변환하면 00001000, 즉 8로 나타내므로 10진수로 -8이라는 올바른 연산결과가 나온다.</p> <p>2) 별첨</p> <p>별첨연산도 덧셈연산과 마찬가지로 2의 보수를 활용하여 연산하는 것이 간편하다. 앞서 말한 양수 + 음수, 음수 + 음수 예제를 통하여 확인해보자.</p> <p>양수 + 음수이고 음수의 크기가 더 큰 경우인 25 - 33을 살펴보자.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>2진수</th> <th>10진수</th> </tr> </thead> <tbody> <tr> <td>MSB 0 0011001 + 1 1011111 ----- 1 1111000</td> <td>25 + (-33) ----- - 8</td> </tr> </tbody> </table> <p>올림수, 즉 캐리가 발생하지 않으면서 올바른 연산결과를 얻을 수 있다.</p> <p>양수 + 음수이고, 양수의 크기가 더 큰 경우 33 - 25의 경우를 살펴보자.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>2진수</th> <th>10진수</th> </tr> </thead> <tbody> <tr> <td>MSB 0 100001 + 1 110011 ----- 1 1001000</td> <td>33 - 25 ----- 8</td> </tr> </tbody> </table> <p style="text-align: right;">제정 연산 71</p>	2진수	10진수	MSB 1 101111 + 0 0011001 ----- 1 1111000	-33 + 25 ----- - 8	2진수	10진수	MSB 0 0011001 + 1 1011111 ----- 1 1111000	25 + (-33) ----- - 8	2진수	10진수	MSB 0 100001 + 1 110011 ----- 1 1001000	33 - 25 ----- 8	25																																								
2진수	10진수																																																						
MSB 1 101111 + 0 0011001 ----- 1 1111000	-33 + 25 ----- - 8																																																						
2진수	10진수																																																						
MSB 0 0011001 + 1 1011111 ----- 1 1111000	25 + (-33) ----- - 8																																																						
2진수	10진수																																																						
MSB 0 100001 + 1 110011 ----- 1 1001000	33 - 25 ----- 8																																																						
수정 사유		오타 수정																																																					
73p	문제-본문	<p style="text-align: center;">독학사 컴퓨터공학과 4단계 통합컴퓨터시스템</p> <div style="border: 1px solid black; padding: 10px; margin: 10px auto; width: fit-content;"> <table style="border-collapse: collapse;"> <tr> <td style="padding: 2px;">0011 (+3)</td> <td style="padding: 2px;">← 피승수</td> </tr> <tr> <td style="padding: 2px;">× 0111 (+7)</td> <td style="padding: 2px;">← 승수</td> </tr> <tr> <td style="padding: 2px;">-----</td> <td></td> </tr> <tr> <td style="padding: 2px;">0011</td> <td style="padding: 2px;">← 피승수 × 0111 = 0011 × 1 × 2⁰</td> </tr> <tr> <td style="padding: 2px;">0011</td> <td style="padding: 2px;">← 피승수 × 0111 = 0011 × 1 × 2¹</td> </tr> <tr> <td style="padding: 2px;">0011</td> <td style="padding: 2px;">← 피승수 × 0111 = 0011 × 1 × 2²</td> </tr> <tr> <td style="padding: 2px;">0000</td> <td style="padding: 2px;">← 피승수 × 0111 = 0011 × 0 × 2³</td> </tr> <tr> <td style="padding: 2px;">-----</td> <td></td> </tr> <tr> <td style="padding: 2px;">0010101 (+21)</td> <td></td> </tr> </table> </div> <p>위의 연산은 사람이 연산을 수행했을 경우이다. 컴퓨터시스템에서의 연산과정은 한 사이클마다 1-shift 연산을 수행할 후 누적부분곱에 부분곱을 더하는 과정을 거친다. 위와 같은 연산은 부분곱의 합을 위하여 2N 비트 덧셈연산이 필요하다. 이 과정을 표로 살펴보자.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>단계</th> <th>연산과정</th> <th>설명</th> </tr> </thead> <tbody> <tr> <td>0</td> <td> <table style="border-collapse: collapse;"> <tr><td style="padding: 2px;">0011</td></tr> <tr><td style="padding: 2px;">× 0111</td></tr> <tr><td style="padding: 2px;">-----</td></tr> <tr><td style="padding: 2px;">0000 0000</td></tr> </table> </td> <td>누적 부분곱 = 초기값 설정</td> </tr> <tr> <td>1</td> <td> <table style="border-collapse: collapse;"> <tr><td style="padding: 2px;">+ 0011</td></tr> <tr><td style="padding: 2px;">-----</td></tr> <tr><td style="padding: 2px;">0000 0011</td></tr> </table> </td> <td>0011 × 1 × 2⁰ 누적 부분곱</td> </tr> <tr> <td>2</td> <td> <table style="border-collapse: collapse;"> <tr><td style="padding: 2px;">+ 0011</td></tr> <tr><td style="padding: 2px;">-----</td></tr> <tr><td style="padding: 2px;">0000 0101</td></tr> </table> </td> <td>0011 × 1 × 2¹ 누적 부분곱</td> </tr> <tr> <td>3</td> <td> <table style="border-collapse: collapse;"> <tr><td style="padding: 2px;">+ 0011</td></tr> <tr><td style="padding: 2px;">-----</td></tr> <tr><td style="padding: 2px;">0001 0101</td></tr> </table> </td> <td>0011 × 1 × 2² 누적 부분곱</td> </tr> <tr> <td>4</td> <td> <table style="border-collapse: collapse;"> <tr><td style="padding: 2px;">+ 0000</td></tr> <tr><td style="padding: 2px;">-----</td></tr> <tr><td style="padding: 2px;">0001 0101</td></tr> </table> </td> <td>0011 × 0 × 2³ 누적 부분곱</td> </tr> </tbody> </table> <p style="text-align: right;">제정 연산 73</p>	0011 (+3)	← 피승수	× 0111 (+7)	← 승수	-----		0011	← 피승수 × 0111 = 0011 × 1 × 2 ⁰	0011	← 피승수 × 0111 = 0011 × 1 × 2 ¹	0011	← 피승수 × 0111 = 0011 × 1 × 2 ²	0000	← 피승수 × 0111 = 0011 × 0 × 2 ³	-----		0010101 (+21)		단계	연산과정	설명	0	<table style="border-collapse: collapse;"> <tr><td style="padding: 2px;">0011</td></tr> <tr><td style="padding: 2px;">× 0111</td></tr> <tr><td style="padding: 2px;">-----</td></tr> <tr><td style="padding: 2px;">0000 0000</td></tr> </table>	0011	× 0111	-----	0000 0000	누적 부분곱 = 초기값 설정	1	<table style="border-collapse: collapse;"> <tr><td style="padding: 2px;">+ 0011</td></tr> <tr><td style="padding: 2px;">-----</td></tr> <tr><td style="padding: 2px;">0000 0011</td></tr> </table>	+ 0011	-----	0000 0011	0011 × 1 × 2 ⁰ 누적 부분곱	2	<table style="border-collapse: collapse;"> <tr><td style="padding: 2px;">+ 0011</td></tr> <tr><td style="padding: 2px;">-----</td></tr> <tr><td style="padding: 2px;">0000 0101</td></tr> </table>	+ 0011	-----	0000 0101	0011 × 1 × 2 ¹ 누적 부분곱	3	<table style="border-collapse: collapse;"> <tr><td style="padding: 2px;">+ 0011</td></tr> <tr><td style="padding: 2px;">-----</td></tr> <tr><td style="padding: 2px;">0001 0101</td></tr> </table>	+ 0011	-----	0001 0101	0011 × 1 × 2 ² 누적 부분곱	4	<table style="border-collapse: collapse;"> <tr><td style="padding: 2px;">+ 0000</td></tr> <tr><td style="padding: 2px;">-----</td></tr> <tr><td style="padding: 2px;">0001 0101</td></tr> </table>	+ 0000	-----	0001 0101	0011 × 0 × 2 ³ 누적 부분곱	<p>+ 0011</p> <p>-----</p> <p>0000 1001</p>
0011 (+3)	← 피승수																																																						
× 0111 (+7)	← 승수																																																						

0011	← 피승수 × 0111 = 0011 × 1 × 2 ⁰																																																						
0011	← 피승수 × 0111 = 0011 × 1 × 2 ¹																																																						
0011	← 피승수 × 0111 = 0011 × 1 × 2 ²																																																						
0000	← 피승수 × 0111 = 0011 × 0 × 2 ³																																																						

0010101 (+21)																																																							
단계	연산과정	설명																																																					
0	<table style="border-collapse: collapse;"> <tr><td style="padding: 2px;">0011</td></tr> <tr><td style="padding: 2px;">× 0111</td></tr> <tr><td style="padding: 2px;">-----</td></tr> <tr><td style="padding: 2px;">0000 0000</td></tr> </table>	0011	× 0111	-----	0000 0000	누적 부분곱 = 초기값 설정																																																	
0011																																																							
× 0111																																																							

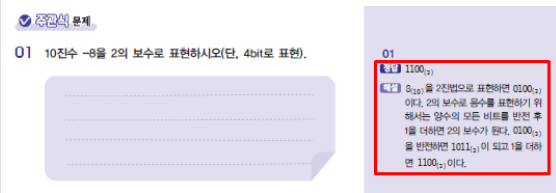
0000 0000																																																							
1	<table style="border-collapse: collapse;"> <tr><td style="padding: 2px;">+ 0011</td></tr> <tr><td style="padding: 2px;">-----</td></tr> <tr><td style="padding: 2px;">0000 0011</td></tr> </table>	+ 0011	-----	0000 0011	0011 × 1 × 2 ⁰ 누적 부분곱																																																		
+ 0011																																																							

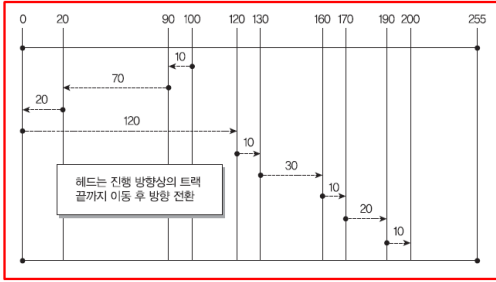
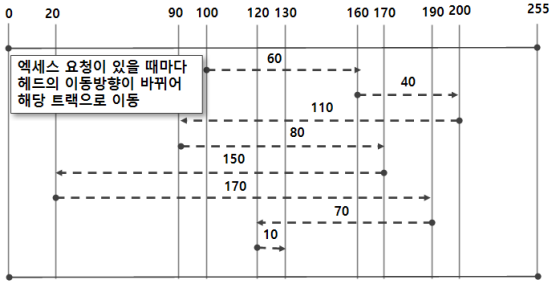
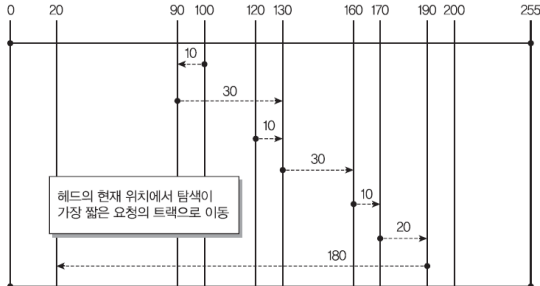
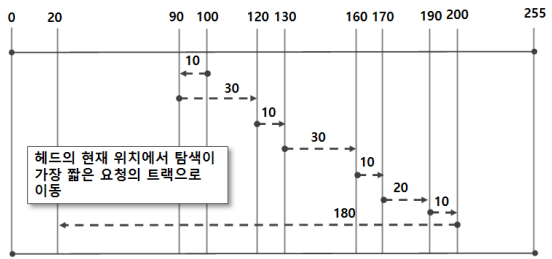
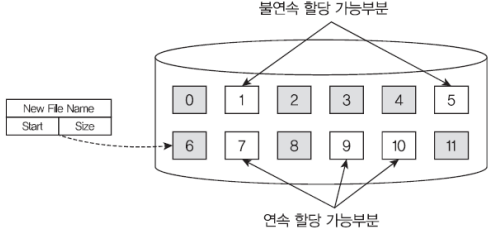
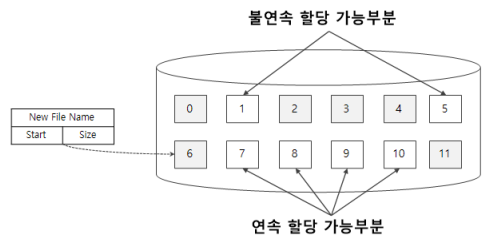
0000 0011																																																							
2	<table style="border-collapse: collapse;"> <tr><td style="padding: 2px;">+ 0011</td></tr> <tr><td style="padding: 2px;">-----</td></tr> <tr><td style="padding: 2px;">0000 0101</td></tr> </table>	+ 0011	-----	0000 0101	0011 × 1 × 2 ¹ 누적 부분곱																																																		
+ 0011																																																							

0000 0101																																																							
3	<table style="border-collapse: collapse;"> <tr><td style="padding: 2px;">+ 0011</td></tr> <tr><td style="padding: 2px;">-----</td></tr> <tr><td style="padding: 2px;">0001 0101</td></tr> </table>	+ 0011	-----	0001 0101	0011 × 1 × 2 ² 누적 부분곱																																																		
+ 0011																																																							

0001 0101																																																							
4	<table style="border-collapse: collapse;"> <tr><td style="padding: 2px;">+ 0000</td></tr> <tr><td style="padding: 2px;">-----</td></tr> <tr><td style="padding: 2px;">0001 0101</td></tr> </table>	+ 0000	-----	0001 0101	0011 × 0 × 2 ³ 누적 부분곱																																																		
+ 0000																																																							

0001 0101																																																							
수정 사유		내용 수정																																																					

위치	오류유형	수정 전	수정 후																																																																
75p 위에서 2~12번째 줄	개념, 공식-설명	<p>부호 없는 수의 나눗셈을 예시로 살펴보자.</p> <p>$153 \div 8$을 보면 $10011001_{(2)} \div 0100_{(2)}$이다. 사람이 연산을 수행했을 경우 다음과 같이 연산을 진행한다.</p> <p>10진수의 나눗셈과 다르지 않다.</p> <table border="1"> <tr><td></td><td></td><td>0010011</td><td>몫</td></tr> <tr><td>제수</td><td>0100</td><td>10011001</td><td>피제수</td></tr> <tr><td></td><td></td><td>100</td><td></td></tr> <tr><td>부분나머지</td><td></td><td>0110</td><td></td></tr> <tr><td></td><td></td><td>0110</td><td></td></tr> <tr><td>부분나머지</td><td></td><td>0100</td><td></td></tr> <tr><td></td><td></td><td>0100</td><td></td></tr> <tr><td></td><td></td><td>0001</td><td>나머지</td></tr> </table>			0010011	몫	제수	0100	10011001	피제수			100		부분나머지		0110				0110		부분나머지		0100				0100				0001	나머지	<p>부호 없는 수의 나눗셈을 예시로 살펴보자.</p> <p>$153 \div 8$을 보면 $10011001_{(2)} \div 1000_{(2)}$이다. 사람이 연산을 수행했을 경우 다음과 같이 연산을 진행한다.</p> <p>10진수의 나눗셈과 다르지 않다.</p> <table border="1"> <tr><td></td><td></td><td>00010011</td><td>몫</td></tr> <tr><td>제수</td><td>1000</td><td>10011001</td><td>피제수</td></tr> <tr><td></td><td></td><td>1000</td><td></td></tr> <tr><td>부분나머지</td><td></td><td>1100</td><td></td></tr> <tr><td></td><td></td><td>1000</td><td></td></tr> <tr><td>부분나머지</td><td></td><td>1001</td><td></td></tr> <tr><td></td><td></td><td>1000</td><td></td></tr> <tr><td></td><td></td><td>1</td><td>나머지</td></tr> </table>			00010011	몫	제수	1000	10011001	피제수			1000		부분나머지		1100				1000		부분나머지		1001				1000				1	나머지
		0010011	몫																																																																
제수	0100	10011001	피제수																																																																
		100																																																																	
부분나머지		0110																																																																	
		0110																																																																	
부분나머지		0100																																																																	
		0100																																																																	
		0001	나머지																																																																
		00010011	몫																																																																
제수	1000	10011001	피제수																																																																
		1000																																																																	
부분나머지		1100																																																																	
		1000																																																																	
부분나머지		1001																																																																	
		1000																																																																	
		1	나머지																																																																
		수정 사유	설명 수정																																																																
77p 제3편- 실제예상문제- 주관식 문제 번호 : 01	정답	 <p>01 10진수 -8을 2의 보수로 표현하시오(단, 4bit로 표현).</p> <p>01 정답 1100₍₂₎ 해설) 8, 2의 보수로 표현하면 0100₍₂₎이다. 2의 보수로 음수를 표현하기 위해서는 양수의 모든 비트를 반전 후 1을 더하면 2의 보수가 된다. 0100₍₂₎을 반전하면 1011₍₂₎이 되고 1을 더하면 1100₍₂₎이다.</p>	<p>정답) 1000₍₂₎ 해설) 10진수 -8은 4bit로 표현 시 범위가 7에서 -8까지이므로, 1000₍₂₎으로 표현한다.</p> <table border="1"> <thead> <tr> <th>10진수</th> <th>2의 보수</th> </tr> </thead> <tbody> <tr><td>-1</td><td>1111</td></tr> <tr><td>-2</td><td>1110</td></tr> <tr><td>-3</td><td>1101</td></tr> <tr><td>-4</td><td>1100</td></tr> <tr><td>-5</td><td>1011</td></tr> <tr><td>-6</td><td>1010</td></tr> <tr><td>-7</td><td>1001</td></tr> <tr><td>-8</td><td>1000</td></tr> </tbody> </table>	10진수	2의 보수	-1	1111	-2	1110	-3	1101	-4	1100	-5	1011	-6	1010	-7	1001	-8	1000																																														
10진수	2의 보수																																																																		
-1	1111																																																																		
-2	1110																																																																		
-3	1101																																																																		
-4	1100																																																																		
-5	1011																																																																		
-6	1010																																																																		
-7	1001																																																																		
-8	1000																																																																		
		수정 사유	정답 및 해설 오류																																																																
219p 제8편- 실제예상문제- 해설&정답 번호 : 14	해설	<p>해설 : FCFS 프로세스를 요청하는 순서대로 프로세서를 할당한다. 따라서 대기시간이 가장 긴 B가 먼저 요청한 작업이므로 B가 가장 먼저 실행된다.</p> <p>정답 : ②</p>	<p>해설 : FCFS는 프로세스를 요청하는 순서대로 할당한다. 따라서 대기시간이 가장 짧은 A 작업이 가장 먼저 실행된다.</p> <p>정답 : ①</p>																																																																
		수정 사유	정답 및 해설 오류																																																																

위치	오류유형	수정 전	수정 후
258p 제10편-제2장- 제2절-1. 탐색 최적화-(1) FCFS	개념,공식-설명	<ul style="list-style-type: none"> Queue : 160, 200, 90, 170, 20, 190, 120, 130 Current head starts at : 100 총 이동거리 : 60 + 40 + 110 + 80 + 150 + 170 + 70 + 10 = 690  <p>[FCFS 풀이]</p>	· 총 이동거리: 60 + 40 + 110 + 80 + 150 + 170 + 70 + 10 = 690 <하단 그림으로 변경> 
		수정 사유	오타 및 그림 교체
259p 제10편-제2장- 제2절-1. 탐색 최적화-(2) SSTF	개념,공식-설명		<하단 그림으로 변경> 
		수정 사유	그림 교체
277p 제10편-제4장- 제4절-1. 디스크 공간 할당 개념	개념,공식-설명		<하단 그림으로 변경> 
		수정 사유	그림 교체

위치	오류유형	수정 전	수정 후								
278p 제10편-제4장- 제4절-3. 디스크 연속 할당 기법	오타	<p>3. 디스크 연속 할당 기법</p> <p>같은 파일에 속한 블록들을 체인으로 연결하기 위해 각 블록에 포인터를 두어 다음 블록 위치를 기록하여 연결하는 기법이다. 블록 단위로 구성되며, 하나의 블록은 여러 개의 섹터로 구성되고, 디렉터리는 파일의 첫 번째 블록을 가리키는 포인터를 포함한다. 여기서 블록은 데이터와 포인터를 지칭하는 용어이다.</p> <p style="text-align: center;">블록단위 연결</p> <p style="text-align: center;">[디스크 연속 할당 기법]</p> <p style="text-align: center;">[디스크 연속 할당 기법 장단점]</p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 50%;">장점</th> <th style="width: 50%;">단점</th> </tr> </thead> <tbody> <tr> <td>·외부 단편화 발생을 방지한다. ·파일 생성 시 공간을 할당하지 않는다. ·파일 크기 변화에 유연하다.</td> <td>·블록마다 포인터 공간이 필요하다. ·포인터 유실 시 파일도 유실된다.</td> </tr> </tbody> </table>	장점	단점	·외부 단편화 발생을 방지한다. ·파일 생성 시 공간을 할당하지 않는다. ·파일 크기 변화에 유연하다.	·블록마다 포인터 공간이 필요하다. ·포인터 유실 시 파일도 유실된다.	<p>3. 디스크 연결 할당 기법</p> <p>같은 파일에 속한 블록들을 체인으로 연결하기 위해 각 블록에 포인터를 두어 다음 블록 위치를 기록하여 연결하는 기법이다. 블록 단위로 구성되며, 하나의 블록은 여러 개의 섹터로 구성되고, 디렉터리는 파일의 첫 번째 블록을 가리키는 포인터를 포함한다. 여기서 블록은 데이터와 포인터를 지칭하는 용어이다.</p> <p style="text-align: center;">블록단위 연결</p> <p style="text-align: center;">[디스크 연결 할당 기법]</p> <p style="text-align: center;">[디스크 연결 할당 기법 장단점]</p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 50%;">장점</th> <th style="width: 50%;">단점</th> </tr> </thead> <tbody> <tr> <td>·외부 단편화 발생을 방지한다. ·파일 생성 시 공간을 할당하지 않는다. ·파일 크기 변화에 유연하다.</td> <td>·블록마다 포인터 공간이 필요하다. ·포인터 유실 시 파일도 유실된다.</td> </tr> </tbody> </table>	장점	단점	·외부 단편화 발생을 방지한다. ·파일 생성 시 공간을 할당하지 않는다. ·파일 크기 변화에 유연하다.	·블록마다 포인터 공간이 필요하다. ·포인터 유실 시 파일도 유실된다.
장점	단점										
·외부 단편화 발생을 방지한다. ·파일 생성 시 공간을 할당하지 않는다. ·파일 크기 변화에 유연하다.	·블록마다 포인터 공간이 필요하다. ·포인터 유실 시 파일도 유실된다.										
장점	단점										
·외부 단편화 발생을 방지한다. ·파일 생성 시 공간을 할당하지 않는다. ·파일 크기 변화에 유연하다.	·블록마다 포인터 공간이 필요하다. ·포인터 유실 시 파일도 유실된다.										
수정 사유		본문 오타									
317p 제12편-제1장- 제2절-2. 4단계 명령어 파이프라인	오타	<p>③ 파이프라인에 의한 속도 공식</p> <p>파이프라인 단계 수 = k이고, 실행할 명령어들의 수 = N일 때 속도 S는 아래와 같다.</p> $S_p = \frac{T_1}{T_k} = \frac{k \times N}{k \times (N-1)}$	<p>③ 파이프라인에 의한 속도 공식</p> <p>파이프라인 단계 수 = k이고, 실행할 명령어들의 수 = N일 때 속도 S는 아래와 같다.</p> $S_p = \frac{T_1}{T_k} = \frac{k \times N}{k \oplus (N-1)}$								
350p 제13편- 실제예상문제- 해설&정답 번호 : 01	문제-본문	<p>01. 임베디드 시스템은 특정 목적이 아닌 범용 목적을 갖는 시스템이다.</p>	<p>01. 임베디드 시스템의 특징으로는 제한적인 기능, 크기에 제약, 저전력, 프로세스와 운영체제의 다양함, 하드디스크 없음이 있다. 높은 성능은 임베디드 시스템의 특징이 아닌 슈퍼컴퓨터 등 높은 연산이 필요한 컴퓨터시스템의 특징이다.</p>								
수정 사유		해설 오류									

도서의 오류로 학습에 불편드린 점 진심으로 사과드립니다.
더 나은 도서를 만들기 위해 노력하는 시대교육그룹이 되겠습니다.