**Computer Architecture Assignment #1**

Your Student Number: 2020320132

Name: 조민규

<<Notice>>

* Please edit the title of this document correctly.
  + (ex. CA1\_2023012345\_Tom-Cruise OR CA1\_2023012345\_홍길동)
* Please write your information(Student number and name) correctly.
* You can write your answers in English or Korean.
* Don’t change the layout(colored in Black) of this document. (Please edit just blue-colored part.)
* Before submitting, don’t forget to convert the file to a PDF format.

**Address 000 | Instruction EA000006 (Example)**

1. Change to binary format: 1110 1010 0000 0000 0000 0000 0000 0110
2. Write assembly code: B #8;
3. Describe why you wrote the assembly code like above:
   1. Type of instruction: According to the figure A3-1 in ARM manual, ‘Branch and branch with link’ is only one instruction set encoding whose values at [25:27] bit is 101. So, I can figure out this instruction is branch instruction.
   2. Operation – Condition Field: According to the A4.1.5(Page A4-10), there is the detail of the branch instruction. ‘Operation’ part of the instruction said that I should check the condition is passed first. The condition field of this instruction is 1110 and it means the instruction can operate unconditionally.
   3. Operation – L: According to page A4-10, branch instruction branches without storing a return address when L is omitted. In the case of this instruction, it doesn’t need to store any return address because the L bit is 0.
   4. Operation – Target Address: According to page A4-10 in ARM manual, the target address is calculated like below.
      1. First, the result of sign-extending the 24-bit signed immediate to 30 bits is 00 0000 0000 0000 0000 0000 0000 0000 0000 0110. (Because the signed immediate is 0000 0000 0000 0000 0000 0110 here.)
      2. Then, get 0000 0000 0000 0000 0000 0000 0001 1000 by shifting the result left two bits.
      3. Because the address of this instruction is 0, the content of PC will be 0 + 8 bytes. So, the target address will be (0+8) + 24 = 32(bytes). It means after the operation of this instruction, PC will be move to 32/4 = 8.
      4. Therefore, I can write the assembly code of this instruction like ‘B #8;’ because the syntax of branch instruction is ‘B{L}{cond} <target\_address>’.
4. What is the meaning of the instruction? : The instruction means ‘branch to address 8’.

Address 001 | Instruction EAFFFFFE

1. Change to binary format: 1110\_1010\_1111\_1111\_1111\_1111\_1111\_1110
2. Write assembly code: B#1
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 101이면 branch명령어이다. 따라서 이 명령어는 branch명령어임을 알 수 있다.  
   또 [24]인 L비트가 0이므로 돌아올 곳을 R14에 저장하는BL명령어가 아닌 B명령어 임을 알 수 있다.(p.160)  
   또 cond비트인 [31:28]이 1110이므로 (p.112)에 따라 항상 분기하라는 명령어이다. 따라서 BAL 명령어 임을 알 수 있다.  
   또 하위 24비트[23:0]은 offset으로 다음 과정을 거쳐pc에 더해진다.  
   1. 30bit로 signed extension한다. -> msb가 1이므로 1을 채운다.  
   ->11/1111/1111/1111/1111/1111/1111/1110   
   2. 2bit left shift 로 singed 32bit로 만든다.-> 1111/1111/1111/1111/1111/1111/1111/1000   
   3. 이 값을 현재 명령어에 8이 더해진 값을 가진pc에 더한다.  
   이 32bit값은 2의 보수를 이용하여 쓰였기 때문에 10진수로 변환하면 -8임을 알 수 있다.  
   따라서 이 명령어의 주소는 (1\*4)=4이므로 현재 pc값은 4+8인 12이다. 따라서 target\_address는 12-8=4이고 이를 4로 나누면 1이다. 따라서 이 명령어는 B #1이다.
4. What is the meaning of the instruction? : 1번째 명령어(address001)를 실행하라는 명령어로 무한루프이다.

Address 002 | Instruction EA0000A7

1. Change to binary format: 1110\_1010\_0000\_0000\_0000\_0000\_1010\_0111
2. Write assembly code: B#171
3. Describe why you wrote the assembly code like above:

A3.1(p.110)에서 [27:25]가 101이면 branch명령어이다. 따라서 이 명령어는 branch명령어임을 알 수 있다.  
또 [24]인 L비트가 0이므로 돌아올 곳을 R14에 저장하는BL명령어가 아닌 B명령어 임을 알 수 있다.(p.160)  
또 cond비트인 [31:28]이 1110이므로 (p.112)에 따라 항상 분기하라는 명령어이다. 따라서 BAL 명령어 임을 알 수 있다.  
또 하위 24비트[23:0]은 offset으로 다음 과정을 거쳐pc에 더해진다.  
1. 30bit로 signed extension한다. -> msb가 0이므로 0을 채운다.  
->00/0000/0000/0000/0000/0000/1010/0111   
2. 2bit left shift 로 singed 32bit로 만든다.-> 0000/0000/0000/0000/0000/0010/1001/1100  
3. 이 값을 현재 명령어에 8이 더해진 값을 가진pc에 더한다.  
이 32bit값은 10진수로 변환하면 668이다.  
따라서 이 명령어의 주소는 (2\*4)=8이므로 현재 pc값은 8+8인 16이다. 따라서 target\_address는 16+668=684이고 이를 4로 나누면 171이다. 따라서 이 명령어는 B #171이다.

1. What is the meaning of the instruction? : 171번째 명령어(address 0AB)로 가라는 뜻

Address 003~005 | Instruction EAFFFFFE

1. Change to binary format: 1110/1010/1111/1111/1111/1111/1111/1110
2. Write assembly code: B#3, B#4, B#5
3. Describe why you wrote the assembly code like above:

A3.1(p.110)에서 [27:25]가 101이면 branch명령어이다. 따라서 이 명령어는 branch명령어임을 알 수 있다.  
또 [24]인 L비트가 0이므로 돌아올 곳을 R14에 저장하는BL명령어가 아닌 B명령어 임을 알 수 있다.(p.160)  
또 cond비트인 [31:28]이 1110이므로 (p.112)에 따라 항상 분기하라는 명령어이다. 따라서 BAL 명령어 임을 알 수 있다.  
또 하위 24비트[23:0]은 offset으로 다음 과정을 거쳐pc에 더해진다.  
1. 30bit로 signed extension한다. -> msb가 1이므로 1을 채운다.  
->11/1111/1111/1111/1111/1111/1111/1110   
2. 2bit left shift 로 singed 32bit로 만든다.-> 1111/1111/1111/1111/1111/1111/1111/1000   
3. 이 값을 현재 명령어에 8이 더해진 값을 가진pc에 더한다.  
이 32bit값은 2의 보수를 이용하여 쓰였기 때문에 10진수로 변환하면 -8임을 알 수 있다.  
따라서 이 명령어의 주소는 (3\*4)=12이므로 현재 pc값은 12+8인 20이다. 따라서 target\_address는 20-8=12이고 이를 4로 나누면 3이다. 따라서 이 명령어는 B #3이다.  
여기서 address 003, 004, 005 로 pc값이 4씩 증가하므로 남은 두 명령어는 B#4, B#5이다.

1. What is the meaning of the instruction? : 이들 모두 자기 자신으로 분기하라는 명령어 이므로 무한루프이다.

Address 006 | Instruction EA0000A4

1. Change to binary format: 1110/1010/0000/0000/0000/0000/1010/0100
2. Write assembly code: B#172
3. Describe why you wrote the assembly code like above:

A3.1(p.110)에서 [27:25]가 101이면 branch명령어이다. 따라서 이 명령어는 branch명령어임을 알 수 있다.  
또 [24]인 L비트가 0이므로 돌아올 곳을 R14에 저장하는BL명령어가 아닌 B명령어 임을 알 수 있다.(p.160)  
또 cond비트인 [31:28]이 1110이므로 (p.112)에 따라 항상 분기하라는 명령어이다. 따라서 BAL 명령어 임을 알 수 있다.  
또 하위 24비트[23:0]은 offset으로 다음 과정을 거쳐pc에 더해진다.  
1. 30bit로 signed extension한다. -> msb가 0이므로 0을 채운다.  
->00/0000/0000/0000/0000/0000/1010/0100   
2. 2bit left shift 로 singed 32bit로 만든다.-> 0000/0000/0000/0000/0000/0010/1001/0000  
3. 이 값을 현재 명령어에 8이 더해진 값을 가진pc에 더한다.  
이 32bit값은 10진수로 변환하면 656이다.  
따라서 이 명령어의 주소는 (6\*4)=24이므로 현재 pc값은 24+8인 32이다. 따라서 target\_address는 32+656=688이고 이를 4로 나누면 172이다. 따라서 이 명령어는 B #172이다.

1. What is the meaning of the instruction? : 172번째 명령어(address 0AC)로 분기하라는 명령어

Address 007 | Instruction EAFFFFFE

1. Change to binary format: 1110/1010/1111/1111/1111/1111/1111/1110
2. Write assembly code: B#7
3. Describe why you wrote the assembly code like above:

A3.1(p.110)에서 [27:25]가 101이면 branch명령어이다. 따라서 이 명령어는 branch명령어임을 알 수 있다.  
또 [24]인 L비트가 0이므로 돌아올 곳을 R14에 저장하는BL명령어가 아닌 B명령어 임을 알 수 있다.(p.160)  
또 cond비트인 [31:28]이 1110이므로 (p.112)에 따라 항상 분기하라는 명령어이다. 따라서 BAL 명령어 임을 알 수 있다.  
또 하위 24비트[23:0]은 offset으로 다음 과정을 거쳐pc에 더해진다.  
1. 30bit로 signed extension한다. -> msb가 1이므로 1을 채운다.

->11/1111/1111/1111/1111/1111/1111/1110   
2. 2bit left shift 로 singed 32bit로 만든다.-> 1111/1111/1111/1111/1111/1111/1111/1000   
3. 이 값을 현재 명령어에 8이 더해진 값을 가진pc에 더한다.  
이 32bit값은 2의 보수를 이용하여 쓰였기 때문에 10진수로 변환하면 -8임을 알 수 있다.  
따라서 이 명령어의 주소는 (7\*4)=28이므로 현재 pc값은 28+8인 36이다. 따라서 target\_address는 36-8=28이고 이를 4로 나누면 7이다. 따라서 이 명령어는 B #7이다.

1. What is the meaning of the instruction? : 7번째 명령어인 자기 자신으로 분기, 무한루프

Address 008 | Instruction E59F2EC8

1. Change to binary format: 1110/0101/1001/1111/0010/1110/1100/1000
2. Write assembly code: LDR r2 [pc, #3784]
3. Describe why you wrote the assembly code like above: YOUR ANSWER  
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 1이므로 load명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r15에 더해 address를 구한다.  
   address=r15 + offset\_12 = r15 + 3784이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.  
   단, 실제 실행시에는 PC에 8을 더한 후 3784를 더해서 address가 계산된다.  
   또 Rd[15:12]는 0010으로 r2를 가리킨다. 따라서 이 명령어는 r2 에 pc+3784에 있는 값을 load하라는 명령어로 ldr r2, [pc, #3784] 이다.
4. What is the meaning of the instruction? : pc에 3784를 더한 주소를 가진 메모리에 들어있는 값을 r2레지스터에 저장한다.

Address 009 | Instruction E3A00040

1. Change to binary format: 1110\_001\_1101\_0\_0000\_0000\_0000\_01000000
2. Write assembly code: MOV r0, #64
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 001이고 [23:23]이 10 이 아니면data processing immediate명령어이다.  
   또 A3.4(p.115)에서 opcode 가 1101이면 MOV 명령어이므로 이 명령어는 MOV명령어이다.  
   또 S비트[20]은 cpsr을 업데이트 할지 정하는 비트인데 0으로 설정되어 있으므로 cpsr을 업데이트 하지 않는다.(p.218)  
   또 Rd field[15:12]가 0000으로 r0를 가리킨다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.  
   Rotate\_imm field[11:8]에 2를 곱한 만큼immed\_8 field[7:0]을 right rotate해서 rd에 들어갈 값을 정하는데 여기선 [11:8]이 0이므로 10진수 64를 나타내는[7:0]이 그대로 들어간다.  
   따라서 이 명령어는 MOV r0, #64이다.
4. What is the meaning of the instruction? R0 레지스터에 10진수 64를 저장하는 명령어이다.

Address 00A | Instruction E5820010

1. Change to binary format: 1110\_0101\_1\_0\_0\_0\_0010\_0000\_000000010000
2. Write assembly code: STR r0, [r2, #16]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 0이므로 store명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r2에 더해 address를 구한다.  
   address=r2 + offset\_12 = r2 + 16이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.   
   또 Rd[15:12]는 0000으로 r0를 가리킨다. 따라서 이 명령어는 r0에 있는 값을r2+16에 store하라는 명령어로 STR r0, [r2,#16] 이다.
4. What is the meaning of the instruction? : 메모리에 r0에 들어있는 값을 r2+16의 주소를 가진 저장한다.

Address 00B | Instruction E5820014

1. Change to binary format: 1110\_0101\_1\_0\_0\_0\_0010\_0000\_000000010100
2. Write assembly code: STR r0, [r2, #20]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 0이므로 store명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r2에 더해 address를 구한다.  
   address=r2 + offset\_12 = r2 + 20이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.   
   또 Rd[15:12]는 0000으로 r0를 가리킨다. 따라서 이 명령어는 r0에 있는 값을r2+20에 store하라는 명령어로 STR r0, [r2,#20] 이다.
4. What is the meaning of the instruction? : 메모리에 r0에 들어있는 값을 r2+20의 주소를 가진 저장한다.

Address 00C | Instruction E5820018

1. Change to binary format: 1110\_ 0101\_1\_0\_0\_0\_0010\_0000\_000000011000
2. Write assembly code: STR r0, [r2, #24]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 0이므로 store명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r2에 더해 address를 구한다.  
   address=r2 + offset\_12 = r2 + 24이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.   
   또 Rd[15:12]는 0000으로 r0를 가리킨다. 따라서 이 명령어는 r0에 있는 값을r2+24에 store하라는 명령어로 STR r0, [r2,#24] 이다.
4. What is the meaning of the instruction? : 메모리에 r0에 들어있는 값을 r2+24의 주소를 가진 저장한다.

Address 00D | Instruction E582001C

1. Change to binary format: 1110\_0101\_1\_0\_0\_0\_0010\_0000\_000000011100
2. Write assembly code: STR r0, [r2, #28]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 0이므로 store명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r2에 더해 address를 구한다.  
   address=r2 + offset\_12 = r2 + 28다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.   
   또 Rd[15:12]는 0000으로 r0를 가리킨다. 따라서 이 명령어는 r0에 있는 값을r2+28에 store하라는 명령어로 STR r0, [r2,#28] 이다.
4. What is the meaning of the instruction? : 메모리에 r0에 들어있는 값을 r+28의 주소를 가진 저장한다.

Address 00E | Instruction E5820020

1. Change to binary format: 1110/0101/1000/0010/0000/0000/0010/0000
2. Write assembly code: STR r0, [r2, #32]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 0이므로 store명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r2에 더해 address를 구한다.  
   address=r2 + offset\_12 = r2 +32이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.   
   또 Rd[15:12]는 0000으로 r0를 가리킨다. 따라서 이 명령어는 r0에 있는 값을r2+32에 store하라는 명령어로 STR r0, [r2,#32] 이다.
4. What is the meaning of the instruction? : 메모리에 r0에 들어있는 값을 r2+32의 주소를 가진 저장한다.

Address 00F | Instruction E5820024

1. Change to binary format: 1110/0101/1000/0010/0000/0000/0010/0100
2. Write assembly code: STR r0, [r2, #36]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 0이므로 store명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r2에 더해 address를 구한다.  
   address=r2 + offset\_12 = r2 + 36이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.   
   또 Rd[15:12]는 0000으로 r0를 가리킨다. 따라서 이 명령어는 r0에 있는 값을r2+36에 store하라는 명령어로 STR r0, [r2,#36] 이다.
4. What is the meaning of the instruction? : 메모리에 r0에 들어있는 값을 r2+36의 주소를 가진 저장한다.

Address 010 | Instruction E3A0003F

1. Change to binary format: 1110/0011/1010/0000/0000/0000/0011/1111
2. Write assembly code: MOV r0, #63
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 001이고 [23:23]이 10 이 아니면data processing immediate명령어이다.  
   또 A3.4(p.115)에서 opcode 가 1101이면 MOV 명령어이므로 이 명령어는 MOV명령어이다.  
   또 S비트[20]은 cpsr을 업데이트 할지 정하는 비트인데 0으로 설정되어 있으므로 cpsr을 업데이트 하지 않는다.(p.218)  
   또 Rd field[15:12]가 0000으로 r0를 가리킨다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.  
   Rotate\_imm field[11:8]에 2를 곱한 만큼immed\_8 field[7:0]을 right rotate해서 rd에 들어갈 값을 정하는데 여기선 [11:8]이 0이므로 10진수 63를 나타내는[7:0]이 그대로 들어간다.  
   따라서 이 명령어는 MOV r0, #63이다.
4. What is the meaning of the instruction? : r0에 63을 저장하는 명령어이다.

Address 011 | Instruction E5820028

1. Change to binary format: 1110/0101/1000/0010/0000/0000/0010/1000
2. Write assembly code: STR r0, [r2, #40]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 0이므로 store명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r2에 더해 address를 구한다.  
   address=r2 + offset\_12 = r2 + 40이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.   
   또 Rd[15:12]는 0000으로 r0를 가리킨다. 따라서 이 명령어는 r0에 있는 값을r2+40에 store하라는 명령어로 STR r0, [r2,#40] 이다.
4. What is the meaning of the instruction? : 메모리에 r0에 들어있는 값을 r2+40의 주소를 가진 저장한다.

Address 012 | Instruction E3A00008

1. Change to binary format: 1110/0011/1010/0000/0000/0000/0000/1000
2. Write assembly code: MOV r0, #8
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 001이고 [23:23]이 10 이 아니면data processing immediate명령어이다.  
   또 A3.4(p.115)에서 opcode 가 1101이면 MOV 명령어이므로 이 명령어는 MOV명령어이다.  
   또 S비트[20]은 cpsr을 업데이트 할지 정하는 비트인데 0으로 설정되어 있으므로 cpsr을 업데이트 하지 않는다.(p.218)  
   또 Rd field[15:12]가 0000으로 r0를 가리킨다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.  
   Rotate\_imm field[11:8]에 2를 곱한 만큼immed\_8 field[7:0]을 right rotate해서 rd에 들어갈 값을 정하는데 여기선 [11:8]이 0이므로 10진수 8을 나타내는[7:0]이 그대로 들어간다.  
   따라서 이 명령어는 MOV r0, #8이다.
4. What is the meaning of the instruction? : r0에 8을 저장하는 명령어이다.

Address 013 | Instruction E582002C

1. Change to binary format: 1110/0101/1000/0010/0000/0000/0010/1100
2. Write assembly code: STR r0, [r2, #44]
3. Describe why you wrote the assembly code like above  
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 0이므로 store명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r2에 더해 address를 구한다.  
   address=r2 + offset\_12 = r2 + 44이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.   
   또 Rd[15:12]는 0000으로 r0를 가리킨다. 따라서 이 명령어는 r0에 있는 값을r2+44에 store하라는 명령어로 STR r0, [r2,#44] 이다
4. What is the meaning of the instruction? : 메모리에 r0에 들어있는 값을 r2+44의 주소를 가진 저장한다.

Address 014 | Instruction E59F3E9C

1. Change to binary format: 1110/0101/1001/1111/0011/1110/1001/1100
2. Write assembly code: LDR r3, [pc, #3740]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 1이므로 load명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r15에 더해 address를 구한다.  
   address=r15 + offset\_12 = r15 + 3740이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.  
   단, 실제 실행시에는 PC에 8을 더한 후 3740를 더해서 address가 계산된다.  
   또 Rd[15:12]는 0011으로 r3를 가리킨다. 따라서 이 명령어는 r3 에 pc+3740에 있는 값을 load하라는 명령어로 LDR r3, [pc, #3740] 이다.
4. What is the meaning of the instruction? : 메모리주소 pc+3740 에 있는 값을 r3 레지스터에 저장하는 명령어이다.

Address 015 | Instruction E59F1E9C

1. Change to binary format: 1110/0101/1001/1111/0001/1110/1001/1100
2. Write assembly code: LDR r1, [pc, #3740]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 1이므로 load명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r15에 더해 address를 구한다.  
   address=r15 + offset\_12 = r15 + 3740이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.  
   단, 실제 실행시에는 PC에 8을 더한 후 3740를 더해서 address가 계산된다.  
   또 Rd[15:12]는 0001으로 r1를 가리킨다. 따라서 이 명령어는 r1 에 pc+3740에 있는 값을 load하라는 명령어로 LDR r1, [pc, #3740] 이다.
4. What is the meaning of the instruction? : 메모리주소 pc+3740 에 있는 값을 r1 레지스터에 저장하는 명령어이다.

Address 016 | Instruction E5831000

1. Change to binary format: 1110/0101/1000/0011/0001/0000/0000/0000
2. Write assembly code: STR r1, [r3, #0]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 0이므로 store명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r3에 더해 address를 구한다.  
   address=r3 + offset\_12 = r3 + 0이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.   
   또 Rd[15:12]는 0001으로 r1를 가리킨다. 따라서 이 명령어는 r1에 있는 값을r3에 store하라는 명령어로 STR r1, [r3,#0] 이다
4. What is the meaning of the instruction? : r1 레지스터에 있는 값을 메모리주소 r3+0 에 저장하는 명령어이다.

Address 017 | Instruction E59F9E98

1. Change to binary format: 1110/0101/1001/1111/1001/1110/1001/1000
2. Write assembly code: LDR r9, [pc, #3736]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 1이므로 load명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r15에 더해 address를 구한다.  
   address=r15 + offset\_12 = r15 + 3736이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.  
   단, 실제 실행시에는 PC에 8을 더한 후 3736를 더해서 address가 계산된다.  
   또 Rd[15:12]는 1001으로 r9를 가리킨다. 따라서 이 명령어는 r9에 pc+3736에 있는 값을 load하라는 명령어로 LDR r9, [pc, #3740] 이다.
4. What is the meaning of the instruction? : 메모리주소 pc+3736 에 있는 값을 r9 레지스터에 저장하는 명령어이다.

Address 018 | Instruction E3A08000

1. Change to binary format: 1110/001/1101/0/0000/1000/0000/0000/0000
2. Write assembly code: MOV r8, #0
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 001이고 [23:23]이 10 이 아니면data processing immediate명령어이다.  
   또 A3.4(p.115)에서 opcode 가 1101이면 MOV 명령어이므로 이 명령어는 MOV명령어이다.  
   또 S비트[20]은 cpsr을 업데이트 할지 정하는 비트인데 0으로 설정되어 있으므로 cpsr을 업데이트 하지 않는다.(p.218)  
   또 Rd field[15:12]가 1000으로 r8를 가리킨다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.  
   Rotate\_imm field[11:8]에 2를 곱한 만큼immed\_8 field[7:0]을 right rotate해서 rd에 들어갈 값을 정하는데 여기선 [11:8]이 0이므로 10진수 0을 나타내는[7:0]이 그대로 들어간다.  
   따라서 이 명령어는 MOV r8, #0이다.
4. What is the meaning of the instruction? : r8에 0을 저장하는 명령어이다.

Address 019 | Instruction E5898000

1. Change to binary format: 1110/0101/1000/1001/1000/0000/0000/0000
2. Write assembly code: STR r8, [r9, #0]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 0이므로 store 명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r9에 더해 address를 구한다.  
   address=r15 + offset\_12 = r9 + 0이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.   
   또 Rd[15:12]는 1000으로 r8를 가리킨다. 따라서 이 명령어는 r9+0에 r8에 있는 값을 store하라는 명령어로 STR r8, [r9, #0] 이다.
4. What is the meaning of the instruction? : 메모리주소r9+0에 r8에 있는 값을 저장하는 명령어이다.

Address 01A | Instruction E5898004

1. Change to binary format: 1110/0101/1000/1001/1000/0000/0000/0100
2. Write assembly code: STR r8, [r9, #4]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 0이므로 store 명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r9에 더해 address를 구한다.  
   address=r15 + offset\_12 = r9 + 4이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.   
   또 Rd[15:12]는 1000으로 r8를 가리킨다. 따라서 이 명령어는 r9+4에 r8에 있는 값을 store하라는 명령어로 STR r8, [r9, #4] 이다.
4. What is the meaning of the instruction? : 메모리주소r9+4에 r8에 있는 값을 저장하는 명령어이다.

Address 01B | Instruction E5898008

1. Change to binary format: 1110/0101/1000/1001/1000/0000/0000/1000
2. Write assembly code: STR r8, [r9, #8]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 0이므로 store 명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r9에 더해 address를 구한다.  
   address=r15 + offset\_12 = r9 +8이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.   
   또 Rd[15:12]는 1000으로 r8를 가리킨다. 따라서 이 명령어는 r9+8에 r8에 있는 값을 store하라는 명령어로 STR r8, [r9, #8] 이다.
4. What is the meaning of the instruction? : 메모리주소r9+8에 r8에 있는 값을 저장하는 명령어이다.

Address 01C | Instruction E589800C

1. Change to binary format: 1110/0101/1000/1001/1000/0000/0000/1100
2. Write assembly code: STR r8, [r9, #12]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 0이므로 store 명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r9에 더해 address를 구한다.  
   address=r15 + offset\_12 = r9 + 12이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.   
   또 Rd[15:12]는 1000으로 r8를 가리킨다. 따라서 이 명령어는 r9+12에 r8에 있는 값을 store하라는 명령어로 STR r8, [r9, #12] 이다.
4. What is the meaning of the instruction? : 메모리주소r9+12에 r8에 있는 값을 저장하는 명령어이다.

Address 01D | Instruction E5898010

1. Change to binary format: 1110/0101/1000/1001/1000/0000/0001/0000
2. Write assembly code: STR r8, [r9, #16]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 0이므로 store명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r9에 더해 address를 구한다.  
   address=r15 + offset\_12 = r9 + 16이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.   
   또 Rd[15:12]는 1000으로 r8를 가리킨다. 따라서 이 명령어는 r9+16에 r8에 있는 값을 store하라는 명령어로 STR r8, [r9, #16] 이다.
4. What is the meaning of the instruction? : : 메모리주소r9+16에 r8에 있는 값을 저장하는 명령어이다.

Address 01E | Instruction E5898014

1. Change to binary format: 1110/0101/1000/1001/1000/0000/0001/0100
2. Write assembly code: STR r8, [r9, #20]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 0이므로 store 명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r9에 더해 address를 구한다.  
   address=r15 + offset\_12 = r9 + 20이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.   
   또 Rd[15:12]는 1000으로 r8를 가리킨다. 따라서 이 명령어는 r9+20에 r8에 있는 값을 store하라는 명령어로 STR r8, [r9, #20] 이다.
4. What is the meaning of the instruction? : 메모리주소 r9+20에 r8에 있는 값을 저장하는 명령어이다.

Address 01F | Instruction E5898018

1. Change to binary format: 1110/0101/1000/1001/1000/0000/0001/1000
2. Write assembly code: STR r8, [r9, #24]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 0이므로 store 명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r9에 더해 address를 구한다.  
   address=r15 + offset\_12 = r9 + 24이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.   
   또 Rd[15:12]는 1000으로 r8를 가리킨다. 따라서 이 명령어는 r9+24에 r8에 있는 값을 store하라는 명령어로 STR r8, [r9, #24] 이다
4. What is the meaning of the instruction? : 메모리주소 r9+24에 r8에 있는 값을 저장하는 명령어이다.

Address 020 | Instruction E59FDE78

1. Change to binary format: 1110/0101/1001/1111/1101/1110/0111/1000
2. Write assembly code: LDR sp, [pc, #3704]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 1이므로 load명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r15에 더해 address를 구한다.  
   address=r15 + offset\_12 = r15 + 3704이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.  
   단, 실제 실행시에는 PC에 8을 더한 후 3784를 더해서 address가 계산된다.  
   또 Rd[15:12]는 1101으로 sp를 가리킨다. 따라서 이 명령어는 sp 에 pc+3704에 있는 값을 load하라는 명령어로 LDR sp, [pc, #3704] 이다.
4. What is the meaning of the instruction? : sp에 메모리주소pc+3704에 있는 값을 저장하라는 명령어이다.

Address 021 | Instruction E5931200

1. Change to binary format: 1110/0101/1001/0011/0001/0010/0000/0000
2. Write assembly code: LDR r1, [r3, #512]
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 010이면 load/store immediate명령어이다. 또 [20]인 L비트가 1이므로 load명령어이다.(p.460).  
   U비트 [23]은 1, B비트[22]는 0이므로 offset\_12[11:0]을 Rn[19:16]인 r3에 더해 address를 구한다.  
   address=r15 + offset\_12 = r3 + 512이다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.  
   단, 실제 실행시에는 PC에 8을 더한 후 512를 더해서 address가 계산된다.  
   또 Rd[15:12]는 0001으로 r1를 가리킨다. 따라서 이 명령어는 r1 에 rc+512에 있는 값을 load하라는 명령어로 LDR r1, [r3, #512] 이다.
4. What is the meaning of the instruction? : r1에 메모리주소 r3+512에 있는 값을 저장하라는 명령어이다.

Address 022 | Instruction E3510001

1. Change to binary format: 1110/001/1010/1/0001/0000/0000/0000/0001
2. Write assembly code: CMP r1, #1
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 001이고 [23:23]이 10 이 아니면data processing immediate명령어이다.  
   또 A3.4(p.115)에서 opcode 가 11010이면 CMP명령어이므로 이 명령어는 CMP명령어이다.  
   또 Rn field[19:16]가 0001으로 r1를 가리킨다.  
   또 cond field인 [31:28]이 1110이므로 (p.112)에 따라 항상 실행하라는 명령어이다.  
   Rotate\_imm field[11:8]에 2를 곱한 만큼immed\_8 field[7:0]을 right rotate해서 rd에 들어갈 값을 정하는데 여기선 [11:8]이 0이므로 10진수 1을 나타내는[7:0]이 그대로 들어간다.  
   따라서 이 명령어는 CMP r1, #1이다.
4. What is the meaning of the instruction? : r1 과 1이 같으면 Z flag 를 set하라는 명령어이다.

Address 023 | Instruction 0A000000

1. Change to binary format: 0000/1010/0000/0000/0000/0000/0000/0000
2. Write assembly code: BEQ #37
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 101이면 branch명령어이다. 따라서 이 명령어는 branch명령어임을 알 수 있다.  
   또 [24]인 L비트가 0이므로 돌아올 곳을 R14에 저장하는BL명령어가 아닌 B명령어 임을 알 수 있다.(p.160)  
   또 cond비트인 [31:28]이 0000이므로 (p.112)에 따라 Z flag가 1이면 분기하라는 명령어이다. 따라서 BEQ 명령어 임을 알 수 있다.  
   또 하위 24비트[23:0]은 offset으로 다음 과정을 거쳐pc에 더해진다.  
   1. 30bit로 signed extension한다. -> msb가 0이므로 0을 채운다.  
   ->00/0000/0000/0000/0000/0000/0000/0000   
   2. 2bit left shift 로 singed 32bit로 만든다.-> 0000/0000/0000/0000/0000/0000/0000/0000  
   3. 이 값을 현재 명령어에 8이 더해진 값을 가진pc에 더한다.  
   이 32bit값은 10진수로 변환하면 0이다.  
   따라서 이 명령어의 주소는 (35\*4)=140이므로 현재 pc값은 140+8인 148이다. 따라서 target\_address는 148+0=148이고 이를 4로 나누면 37이다. 따라서 이 명령어는 B #37이다
4. What is the meaning of the instruction? : z flag가 set되어있으면 37번째 명령어(address 025)로 분기하라는 명령어이다.

Address 024 | Instruction EAFFFFFB

1. Change to binary format: 1110/1010/1111/1111/1111/1111/1111/1011
2. Write assembly code: B #33
3. Describe why you wrote the assembly code like above:   
   A3.1(p.110)에서 [27:25]가 101이면 branch명령어이다. 따라서 이 명령어는 branch명령어임을 알 수 있다.  
   또 [24]인 L비트가 0이므로 돌아올 곳을 R14에 저장하는BL명령어가 아닌 B명령어 임을 알 수 있다.(p.160)  
   또 cond비트인 [31:28]이 1110이므로 (p.112)에 따라 항상 분기하라는 명령어이다. 따라서 BAL 명령어 임을 알 수 있다.  
   또 하위 24비트[23:0]은 offset으로 다음 과정을 거쳐pc에 더해진다.  
   1. 30bit로 signed extension한다. -> msb가 1이므로 1을 채운다.  
   ->11/1111/1111/1111/1111/1111/1111/1011  
   2. 2bit left shift 로 singed 32bit로 만든다.-> 1111/1111/1111/1111/1111/1111/1110/1100  
   3. 이 값을 현재 명령어에 8이 더해진 값을 가진pc에 더한다.  
   이 32bit값은 10진수로 변환하면 -20이다.  
   따라서 이 명령어의 주소는 (36\*4)=144이므로 현재 pc값은 144+8인 152이다. 따라서 target\_address는 152-20=132이고 이를 4로 나누면 33이다. 따라서 이 명령어는 B #33이다
4. What is the meaning of the instruction? : 33번째 명령어(address021)로 분기하라는 명령어이다.

Explain the actual execution flow of the instructions(Address 000~024)

000| b#8: pc가 8로 설정되어 8번째 명령어로 분기함  
008|ldr r2, [pc, #3784]: pc에 3784를 더한 주소로 접근해 해당 값을 r2에 저장함. 단, pc는 현재 명령어의 위치 008에 8을 더한 0x10을 나타냄.  
009| mov r0, #64: r0에 64를 저장.  
00A|str r0, [r2, #16]: mem[r2+16]에 r0(64)를 저장.  
00B|str r0, [r2, #20]: mem[r2+20]에 r0(64)를 저장.  
00C|str r0, [r2, #24] : mem[r2+24]에 r0(64)를 저장.  
00D|str r0, [r2, #28] : mem[r2+28]에 r0(64)를 저장.  
00E|str r0, [r2, #32] : mem[r2+32]에 r0(64)를 저장.  
00F|str r0, [r2, #36] : mem[r2+36]에 r0(64)를 저장.  
010|mov r0, #63 : r0에 63을 저장  
011|str r0, [r2,#40] : mem[r2+40]에 r0(64)를 저장.  
012|mov r0, #8 : r0에 8을 저장  
013|str r0, [r2, #44] : mem[r2+44]에 r0(8)을 저장  
014|ldr r3, [pc, #3740] : r3에 mem[pc+3740]을 저장. 단 pc는 현재 명령어의 주소에서 8을 더한 값을 나타냄.  
015|ldr r1, [pc, #3740] : r1에 mem[pc+3740]을 저장. 단 pc는 현재 명령어의 주소에서 8을 더한 값을 나타냄.  
016|str r1, [r3, #0] : mem[r3+0]에 r1(unknown)을 저장  
017|str r9, [pc, #3736] : mem[pc+3736]에 r9을 저장. 단 pc는 현재 명령어의 주소에서 8을 더한 값을 나타냄.  
018|mov r8, #0 : r8에 0을 저장  
019|str r8, [r9, #0] : mem[r9+0]에r8(0)을 저장  
01A|str r8, [r9, #4] : mem[r9+4]에r8(0)을 저장  
01B|str r8, [r9, #8] : mem[r9+8]에r8(0)을 저장  
01C|str r8, [r9, #12] : mem[r9+12]에r8(0)을 저장  
01D|str r8, [r9, #16] : mem[r9+16]에r8(0)을 저장  
01E|str r8, [r9, #20] : mem[r9+20]에r8(0)을 저장  
01F|str r8, [r9, #24] : mem[r9+24]에r8(0)을 저장  
020:str sp, [pc, #3704]: mem[pc+3704]에 sp(r13)(unknown)을 저장  
021|str r1, [r3, #512] : mem[r3+512]에 r1(unknown)을 저장  
022|cmp r1, #1 : r1(unknown)과 상수1이 같으면 z-flag=1 else z-flag=0  
023|beq #37 : z-flag=1 이면 37(address025)번째 명령어로 분기 else pc=pc+4  
<case1>(r1=1)  
pc가 0x25가 되어 더 이상 읽을 명령어가 없어 pc는 계속 증가하나 명령어를 실행하진 못한다.  
<case2>(r1!=1)  
024| b#33: 33번째 명령어(address021)로 분기  
021| str r1, [r3, #512] : mem[r3+512]에 r1(not 1)을 저장  
022|cmp r1, #1 : r1(not 1)과 상수1이 다르므로 z-flag=0  
023|beq #37 : z-flag=0 이므로 pc=pc+4  
(이하 무한루프)

Specify where the execution ends (If not, specify the range repeated in detail)

case1에서는 별다른 연산없이 pc만 계속 증가한다.  
case2 에서는 address021~address024를 무한루프를 돈다.