**Computer Architecture Assignment #1**

**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: 11101010111111111111111111111110
2. Write assembly code: jmp far FFFFFFFEh
3. Describe why you wrote the assembly code like above: transfers control to the instruction at the address specified by the operand. The operand in this case is FAR FFFFFFFEh
4. What is the meaning of the instruction? : to perform a far jump to the address FFFFFFFEh, which is the highest possible address in the 32-bit address space

Address 002 | Instruction EA0000A7

1. Change to binary format: 11101010000000000000000010100111
2. Write assembly code: jmp near A70000h
3. Describe why you wrote the assembly code like above: transfers control to the instruction at the address specified by the operand. The operand in this case is NEAR A70000h
4. What is the meaning of the instruction? : to perform a near jump to the address A70000h, which is a memory address in the 32-bit address space

Address 003~005 | Instruction EAFFFFFE

1. Change to binary format: 11101010111111111111111111111110
2. Write assembly code: jmp far dword ptr [005h]
3. Describe why you wrote the assembly code like above: transfers control to the instruction at the address specified by the operand. In this case, the operand is stored at memory address 005h
4. What is the meaning of the instruction? : The meaning of the instruction is to perform a far jump to the address specified by the operand, which is a double word (4 bytes) stored at memory address 005h

Address 006 | Instruction EA0000A4

1. Change to binary format: 11101010000000000000000010100100
2. Write assembly code: jmp near A40000h
3. Describe why you wrote the assembly code like above: transfers control to the instruction at the address specified by the operand. The operand in this case is NEAR A40000h, which specifies the address to jump to
4. What is the meaning of the instruction? : to perform a near jump to the address A40000h, which is a memory address in the 32-bit address space

Address 007 | Instruction EAFFFFFE

1. Change to binary format: 11101010111111111111111111111110
2. Write assembly code: jmp far dword ptr [FFFFh]
3. Describe why you wrote the assembly code like above: transfers control to the instruction at the address specified by the operand. In this case, the operand is stored at memory address FFFFh
4. What is the meaning of the instruction? : to perform a far jump to the address specified by the operand, which is a double word (4 bytes) stored at memory address FFFFh

Address 008 | Instruction E59F2EC8

1. Change to binary format: 11100101001111110010111011001000
2. Write assembly code: ldr r2, [pc, #-236]!
3. Describe why you wrote the assembly code like above: it loads a number into register r2 from memory at a location 236 bytes prior to the program counter (pc). After the value is loaded, the exclamation point at the end of the memory address instructs the computer to change its pointer to the memory location
4. What is the meaning of the instruction? : to load a value from a memory address that is 236 bytes before the current program counter (pc), and store it in register r2. The memory address is specified using a combination of the pc register and an offset of -236

Address 009 | Instruction E3A00040

1. Change to binary format: 11100011101000000000000001000000
2. Write assembly code: mov r0, #64
3. Describe why you wrote the assembly code like above: an ARM assembly code instruction. The ARM architecture is a RISC (Reduced Instruction Set Computing) architecture used in many embedded systems. The instruction is a MOV (move) instruction that moves an immediate value into a register. In this case, it moves the decimal value 64 (or 0x40 in hexadecimal) into register r0
4. What is the meaning of the instruction? : to move the immediate value 64 (0x40 in hexadecimal) into register r0. The MOV instruction is commonly used in ARM assembly code to move data between registers or between registers and memory

Address 00A | Instruction E5820010

1. Change to binary format: 11100101000000100000000000010000
2. Write assembly code: str r0, [r2, #16]
3. Describe why you wrote the assembly code like above: it stores the value of register r0 into memory at the address specified by the sum of the value of register r2 and the immediate value 16
4. What is the meaning of the instruction? : to store the value of register r0 into memory at the address specified by the sum of the value of register r2 and the immediate value 16. The STR instruction is commonly used in ARM assembly code to store data from registers into memory for later use or processing

Address 00B | Instruction E5820014

1. Change to binary format: 11100101000000100000000000010100
2. Write assembly code: str r0, [r2, #20]
3. Describe why you wrote the assembly code like above: it stores the value of register r0 into memory at the address specified by the sum of the value of register r2 and the immediate value 20
4. What is the meaning of the instruction? : to store the value of register r0 into memory at the address specified by the sum of the value of register r2 and the immediate value 20. The STR instruction is commonly used in ARM assembly code to store data from registers into memory for later use or processing

Address 00C | Instruction E5820018

1. Change to binary format: 11100101100000100000000000011000
2. Write assembly code: str r0, [r2, #0x18]
3. Describe why you wrote the assembly code like above: it stores the value of register r0 into memory at the address specified by the sum of the value of register r2 and the immediate value 18
4. What is the meaning of the instruction? : to store the value of register r0 into memory at the address specified by the sum of the value of register r2 and the immediate value 18. The STR instruction is commonly used in ARM assembly code to store data from registers into memory for later use or processing

Address 00D | Instruction E582001C

1. Change to binary format: 11100101100000100000000000011100
2. Write assembly code: str r0, [r2, #0x1c]
3. Describe why you wrote the assembly code like above: it stores the value of register r0 into memory at the address specified by the sum of the value of register r2 and the immediate value 1c
4. What is the meaning of the instruction? : to store the value of register r0 into memory at the address specified by the sum of the value of register r2 and the immediate value 1c. The STR instruction is commonly used in ARM assembly code to store data from registers into memory for later use or processing

Address 00E | Instruction E5820020

1. Change to binary format: 11100101100000100000000000100000
2. Write assembly code: str r0, [r2, #0x20]
3. Describe why you wrote the assembly code like above: it stores the value of register r0 into memory at the address specified by the sum of the value of register r2 and the immediate value 20
4. What is the meaning of the instruction? : to store the value of register r0 into memory at the address specified by the sum of the value of register r2 and the immediate value 20. The STR instruction is commonly used in ARM assembly code to store data from registers into memory for later use or processing

Address 00F | Instruction E5820024

1. Change to binary format: 11100101100000100000000000100100
2. Write assembly code: str r0, [r2, #0x24]
3. Describe why you wrote the assembly code like above: it stores the value of register r0 into memory at the address specified by the sum of the value of register r2 and the immediate value 24
4. What is the meaning of the instruction? : to store the value of register r0 into memory at the address specified by the sum of the value of register r2 and the immediate value 24. The STR instruction is commonly used in ARM assembly code to store data from registers into memory for later use or processing

Address 010 | Instruction E3A0003F

1. Change to binary format: 11100011101000000000000000111111
2. Write assembly code: mov r0, #0x3f
3. Describe why you wrote the assembly code like above: an ARM assembly code instruction. The ARM architecture is a RISC (Reduced Instruction Set Computing) architecture used in many embedded systems. The instruction is a MOV (move) instruction that moves an immediate value into a register. In this case, it moves the decimal value 63 (or 0x3f in hexadecimal) into register r0
4. What is the meaning of the instruction? : to move the immediate value 63 (0x3f in hexadecimal) into register r0. The MOV instruction is commonly used in ARM assembly code to move data between registers or between registers and memory

Address 011 | Instruction E5820028

1. Change to binary format: 11100101100000100000000000101000
2. Write assembly code: str r0, [r2, #0x28]
3. Describe why you wrote the assembly code like above: it stores the value of register r0 into memory at the address specified by the sum of the value of register r2 and the immediate value 28
4. What is the meaning of the instruction? : to store the value of register r0 into memory at the address specified by the sum of the value of register r2 and the immediate value 28. The STR instruction is commonly used in ARM assembly code to store data from registers into memory for later use or processing

Address 012 | Instruction E3A00008

1. Change to binary format: 11100011101000000000000000001000
2. Write assembly code: mov r0, #8
3. Describe why you wrote the assembly code like above: an ARM assembly code instruction. The ARM architecture is a RISC (Reduced Instruction Set Computing) architecture used in many embedded systems. The instruction is a MOV (move) instruction that moves an immediate value into a register. In this case, it moves the decimal value 8 (or 0x08 in hexadecimal) into register r0
4. What is the meaning of the instruction? : to move the immediate value 8 (0x08 in hexadecimal) into register r0. The MOV instruction is commonly used in ARM assembly code to move data between registers or between registers and memory

Address 013 | Instruction E582002C

1. Change to binary format: 11100101100000100000000000101100
2. Write assembly code: str r0, [r2, #0x2c]
3. Describe why you wrote the assembly code like above: it stores the value of register r0 into memory at the address specified by the sum of the value of register r2 and the immediate value 2c
4. What is the meaning of the instruction? : to store the value of register r0 into memory at the address specified by the sum of the value of register r2 and the immediate value 2c. The STR instruction is commonly used in ARM assembly code to store data from registers into memory for later use or processing

Address 014 | Instruction E59F3E9C

1. Change to binary format: 11100101100111110011111010011100
2. Write assembly code: ldr r3, [pc, #0xe9c]
3. Describe why you wrote the assembly code like above: it loads a number into register r3 from memory at a location e9c bytes prior to the program counter (pc). After the value is loaded, the exclamation point at the end of the memory address instructs the computer to change its pointer to the memory location
4. What is the meaning of the instruction? : to load a value from a memory address that is e9c bytes before the current program counter (pc), and store it in register r3. The memory address is specified using a combination of the pc register and an offset of e9c

Address 015 | Instruction E59F1E9C

1. Change to binary format: 11100101100111110001111010011100
2. Write assembly code: ldr r1, [pc, #0xe9c]
3. Describe why you wrote the assembly code like above: it loads a number into register r1 from memory at a location e9c bytes prior to the program counter (pc). After the value is loaded, the exclamation point at the end of the memory address instructs the computer to change its pointer to the memory location
4. What is the meaning of the instruction? : to load a value from a memory address that is e9c bytes before the current program counter (pc), and store it in register r1. The memory address is specified using a combination of the pc register and an offset of e9c

Address 016 | Instruction E5831000

1. Change to binary format: 11100101100000110001000000000000
2. Write assembly code: str r1, [r3]
3. Describe why you wrote the assembly code like above: it stores the value of register r1 into memory at the address specified by the sum of the value of register r3
4. What is the meaning of the instruction? : to store the value of register r1 into memory at the address specified by the sum of the value of register r3. The STR instruction is commonly used in ARM assembly code to store data from registers into memory for later use or processing

Address 017 | Instruction E59F9E98

1. Change to binary format: 11100101100111111001111010011000
2. Write assembly code: ldr sb, [pc, #0xe98]
3. Describe why you wrote the assembly code like above: it loads a number into register sb from memory at a location e98 bytes prior to the program counter (pc). After the value is loaded, the exclamation point at the end of the memory address instructs the computer to change its pointer to the memory location
4. What is the meaning of the instruction? : to load a value from a memory address that is e98 bytes before the current program counter (pc), and store it in register sb. The memory address is specified using a combination of the pc register and an offset of e98

Address 018 | Instruction E3A08000

1. Change to binary format: 11100011101000001000000000000000
2. Write assembly code: mov r8, #0
3. Describe why you wrote the assembly code like above: an ARM assembly code instruction. The ARM architecture is a RISC (Reduced Instruction Set Computing) architecture used in many embedded systems. The instruction is a MOV (move) instruction that moves an immediate value into a register. In this case, it moves the decimal value 0 (or 0x0 in hexadecimal) into register r8
4. What is the meaning of the instruction? : to move the immediate value 0 (0x0 in hexadecimal) into register r8. The MOV instruction is commonly used in ARM assembly code to move data between registers or between registers and memory

Address 019 | Instruction E5898000

1. Change to binary format: 11100101100010011000000000000000
2. Write assembly code: str r8, [sb]
3. Describe why you wrote the assembly code like above: it stores the value of register r8 into memory at the address specified by the sum of the value of register sb
4. What is the meaning of the instruction? : to store the value of register r8 into memory at the address specified by the sum of the value of register sb. The STR instruction is commonly used in ARM assembly code to store data from registers into memory for later use or processing

Address 01A | Instruction E5898004

1. Change to binary format: 11100101100010011000000000000100
2. Write assembly code: str r8, [sb, #4]
3. Describe why you wrote the assembly code like above: it stores the value of register r8 into memory at the address specified by the sum of the value of register sb and the immediate value 4
4. What is the meaning of the instruction? : to store the value of register r8 into memory at the address specified by the sum of the value of register sb and the immediate value 4. The STR instruction is commonly used in ARM assembly code to store data from registers into memory for later use or processing

Address 01B | Instruction E5898008

1. Change to binary format: 11100101100010011000000000001000
2. Write assembly code: str r8, [sb, #8]
3. Describe why you wrote the assembly code like above: it stores the value of register r8 into memory at the address specified by the sum of the value of register sb and the immediate value 8
4. What is the meaning of the instruction? : to store the value of register r8 into memory at the address specified by the sum of the value of register sb and the immediate value 8. The STR instruction is commonly used in ARM assembly code to store data from registers into memory for later use or processing

Address 01C | Instruction E589800C

1. Change to binary format: 11100101100010011000000000001100
2. Write assembly code: str r8, [sb, #0xc]
3. Describe why you wrote the assembly code like above: it stores the value of register r8 into memory at the address specified by the sum of the value of register sb and the immediate value c
4. What is the meaning of the instruction? : to store the value of register r8 into memory at the address specified by the sum of the value of register sb and the immediate value c. The STR instruction is commonly used in ARM assembly code to store data from registers into memory for later use or processing

Address 01D | Instruction E5898010

1. Change to binary format: 11100101100010011000000000010000
2. Write assembly code: str r8, [sb, #0x10]
3. Describe why you wrote the assembly code like above: it stores the value of register r8 into memory at the address specified by the sum of the value of register sb and the immediate value 10
4. What is the meaning of the instruction? : to store the value of register r8 into memory at the address specified by the sum of the value of register sb and the immediate value 10. The STR instruction is commonly used in ARM assembly code to store data from registers into memory for later use or processing

Address 01E | Instruction E5898014

1. Change to binary format: 11100101100010011000000000010100
2. Write assembly code: str r8, [sb, #0x14]
3. Describe why you wrote the assembly code like above: it stores the value of register r8 into memory at the address specified by the sum of the value of register sb and the immediate value 14
4. What is the meaning of the instruction? : to store the value of register r8 into memory at the address specified by the sum of the value of register sb and the immediate value 14. The STR instruction is commonly used in ARM assembly code to store data from registers into memory for later use or processing

Address 01F | Instruction E5898018

1. Change to binary format: 11100101100010011000000000011000
2. Write assembly code: str r8, [sb, #0x18]
3. Describe why you wrote the assembly code like above: it stores the value of register r8 into memory at the address specified by the sum of the value of register sb and the immediate value 18
4. What is the meaning of the instruction? : to store the value of register r8 into memory at the address specified by the sum of the value of register sb and the immediate value 18. The STR instruction is commonly used in ARM assembly code to store data from registers into memory for later use or processing

Address 020 | Instruction E59FDE78

1. Change to binary format: 11100101100111111101111001111000
2. Write assembly code: ldr sp, [pc, #0xe78]
3. Describe why you wrote the assembly code like above: it loads a number into register sp from memory at a location e9c bytes prior to the program counter (pc). After the value is loaded, the exclamation point at the end of the memory address instructs the computer to change its pointer to the memory location
4. What is the meaning of the instruction? : to load a value from a memory address that is e78 bytes before the current program counter (pc), and store it in register sp. The memory address is specified using a combination of the pc register and an offset of e78

Address 021 | Instruction E5931200

1. Change to binary format: 11100101100100110001001000000000
2. Write assembly code: ldr r1, [r3, #0x200]
3. Describe why you wrote the assembly code like above: it loads a number into register r1 from memory at a location 200 bytes prior to r3. After the value is loaded, the exclamation point at the end of the memory address instructs the computer to change its pointer to the memory location
4. What is the meaning of the instruction? : to load a value from a memory address that is 200 bytes before r3 and store it in register r1. The memory address is specified using a combination of the pc register and an offset of 200

Address 022 | Instruction E3510001

1. Change to binary format: 11100011010100010000000000000001
2. Write assembly code: cmp r1, #1
3. Describe why you wrote the assembly code like above: When the processor executes a conditional instruction, it checks the status flags register and jumps to the target label if it meets the conditions, otherwise falls through to the next instruction. compare the value of r1 and value 1
4. What is the meaning of the instruction? : compare the value of r1 and value 1

Address 023 | Instruction 0A000000

1. Change to binary format: 00001010000000000000000000000000
2. Write assembly code: beq #8
3. Describe why you wrote the assembly code like above: When the processor executes a branch equal instruction, it checks the status flags register and jumps to the target label if it meets the conditions, otherwise falls through to the next instruction.
4. What is the meaning of the instruction? : branch if equal 8 and the value in register in step before (address 22)

Address 024 | Instruction EAFFFFFB

1. Change to binary format: 11101010111111111111111111111011
2. Write assembly code: b #0xfffffff4
3. Describe why you wrote the assembly code like above: When the processor executes a branch instruction, it checks the status flags register and jumps to the target label if it meets the conditions, otherwise falls through to the next instruction.
4. What is the meaning of the instruction? : branch to memory address of #0xfffffff4. it works by command address 023

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

001 -> 003 -> 002 -> 006 -> 007-> 005 -> 004 -> 008 -> 009 -> 00A -> 00B -> 00C -> 00D -> 00E -> 00F -> 010 -> 011 -> 012 -> 013 -> 014 -> 015 -> 016 -> 017 -> 018 -> 019 -> 01A -> 01B -> 01C -> 01D -> 01E -> 01F -> 020 -> 021 -> 022 -> 023 -> 024

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

Address 024 is the branch to 0xfffffff4 so if the condition is equal program can be jump to 0xfffffff4 and successfully done with this program, otherwise, more register information or address cannot access in this situation so is not equal, cannot determine.