23122013 - BTTL3

23122013 - Đinh Đức Tài, BTTL3 - HTMT - 23TNT1

Câu 1:

Chuyển sang chương trình C cho đoạn mã bên dưới. Biết biến i, result, địa chỉ cơ sở của mảng A lần lượt chứa ở thanh ghi X10, X9, X1. Giải thích vai trò của từng dòng lệnh trong đoạn mã LEGv8.

Address	Instruction
10000	ORR X10, XZR, XZR
10004	LOOP: LDUR X11, [X1, #0]
10008	ADD X0, X0, X11
10012	ADDI X1, X1, #8
10016	ADDI X10, X10, #1
10020	SUBIS XZR, X10, 100
10024	B.LT LOOP

```
long long result = 0;
int i;

for (i = 0; i < 100; i++) {
  result = result + A[i];
}
```

- ORR X10, XZR, XZR : **Khởi tạo i = 0.** OR logic giữa thanh ghi zero (XZR) với chính nó và lưu kết quả vào X10. X10 chứa biến i.
- LOOP: LDUR X11, [X1, #0] : **Tải giá trị A[i] vào thanh ghi tạm (X11).** LDUR tải [X1 + 0] vào thanh ghi X11. X1 là địa chỉ của A[i] (được tăng dần từ địa chỉ cơ sở sau mỗi vòng lặp)
- ADD X0, X0, X11 : **Cộng dồn vào result.** Cộng X11 (tức A[i]) vào thanh ghi X0 và lưu lại kết quả vào X0 (tức result).
- ADDI X1, X1, #8 : **Di chuyển con trỏ đến phần tử tiếp theo.** Cộng địa chỉ cơ sở X1 với 8 (double-word).
- ADDI X10, X10, #1 : **Tăng biến đếm i lên 1 (i++)**. Cộng giá trị trong X10 với 1.
- SUBIS XZR, X10, 100 : **So sánh i với 100.** Trừ X10 cho 100 (i - 100) và cập nhật các cờ trạng thái (N, Z, V, C).
- B.LT LOOP : **Kiểm tra điều kiện lặp.** B.LT sẽ nhảy về nhãn LOOP nếu kết quả của phép SUBIS trước đó là số âm (i < 100).

Câu 2:

Đoạn mã C thực hiện việc điền vào một mảng 10 phần tử, với mỗi phần tử array[i] bằng giá trị lớn hơn giữa i và num (biến num này giảm dần sau mỗi vòng lặp).

Instruction

```
int array[10];
void main () {
      int num;
      set_array(num);
}
void set_array (int num) {
      for (int i=0; i<10; i++) {
            array[i] = compare(i,num);
            num--;
      }
}
int compare (int a, int b) {
      if (sub(b,a) >= 0)
            return b;
      else
            return a;
}
int sub (int a, int b) {
      return a-b;
}
```

```
// Giả định thanh ghi X19 chứa địa chỉ cơ sở của 'array'
// Giả định thanh ghi X21 chứa giá trị ban đầu của 'num'
// Thanh ghi X20 sẽ được dùng cho biến 'i'

set_array:
    // --- Prologue: Lưu các thanh ghi cần dùng lên stack ---
    SUBI SP, SP, #24          // Cấp phát 24 byte trên stack
    STUR LR, [SP, #16]        // Lưu thanh ghi Link Register (địa chỉ trả
về)
    STUR X20, [SP, #8]        // Lưu X20 (sẽ dùng cho 'i')
    STUR X21, [SP, #0]        // Lưu X21 (sẽ dùng cho 'num')

    MOV X21, X0               // Lấy tham số 'num' từ X0 vào X21
    MOV X20, XZR              // Khởi tạo i = 0 (X20 = 0)

loop:
    // --- Điều kiện lặp: for (i=0; i<10; i++) ---
    SUBI X9, X20, #10         // So sánh i với 10
    B.GE end_loop             // Nếu i >= 10, thoát vòng lặp

    // --- Chuẩn bị gọi hàm compare(i, num) ---
    MOV X0, X20               // Đặt đối số đầu tiên a = i (vào X0)
    MOV X1, X21               // Đặt đối số thứ hai b = num (vào X1)
```

```
    BL compare                // Gọi hàm compare, kết quả trả về trong X0

    // --- Gán kết quả: array[i] = kết quả ---
    LSL X9, X20, #3           // Tính offset: X9 = i * 8 (byte offset)
    ADD X9, X19, X9           // Tính địa chỉ của array[i]: base + offset
    STUR X0, [X9, #0]         // Lưu kết quả từ compare vào array[i]

    // --- Cập nhật biến ---
    SUBI X21, X21, #1         // num--
    ADDI X20, X20, #1         // i++
    B loop                    // Quay lại đầu vòng lặp

end_loop:
    // --- Epilogue: Khôi phục các thanh ghi từ stack ---
    LDUR X21, [SP, #0]        // Khôi phục X21
    LDUR X20, [SP, #8]        // Khôi phục X20
    LDUR LR, [SP, #16]        // Khôi phục Link Register
    ADDI SP, SP, #24          // Giải phóng stack
    BR LR                     // Trở về nơi đã gọi

// --- Hàm compare(int a, int b) ---
compare:
    // So sánh a (X0) và b (X1) để trả về max(a,b)
    SUB X9, X1, X0            // X9 = b - a
    B.GE return_b             // Nếu b - a >= 0 (b >= a), nhảy tới return_b
    // else, return a (giá trị của a vẫn đang ở trong X0)
    BR LR                     // Trả về a (trong X0)
return_b:
    MOV X0, X1                // Gán X0 = X1 để trả về b
    BR LR                     // Trả về b (trong X0)

sub:
    SUB X0, X0, X1            // return a - b
    BR LR
```

Câu 3:

Chuyển sang LEGv8 cho đoạn mã bên dưới. Biết biến i, num, địa chỉ cơ sở của mảng array lần lượt nằm ở thanh ghi X20, X21, X19. Giải thích vai trò của từng dòng lệnh trong đoạn mã

LEGv8.

Address	Instruction
0x00400030	Main: ADDI X0, XZR, #10
0x00400034	BL Function
0x00400038	ADD X19, X0, XZR
0x0040003C	ADDI X8, #1 // syscall number for exit
0x00400040	ADDI X0, #0 // return 0 status
0x00400044	SVC #0 // invoke syscall to exit
0x00400048	Func: SUBI SP, SP, #32
0x0040004C	STUR X19, [SP, #24]
0x00400050	STUR X20, [SP, #16]
0x00400054	STUR LR, [SP, #8]
0x00400058	STUR X0, [SP, #0]
0x0040005C	ADDI X19, XZR, #0
0x00400060	ADDI X20, XZR, #0
0x00400064	Loop: SUBS XZR, X19, X0
0x00400068	B.LT L1
0x0040006C	B FinalFunction
0x00400070	L1: ADDI X0, X20, #0
0x00400074	BL Check
0x00400078	CBZ X0, Inc_Loop
0x0040007C	ADD X19, X19, X20
0x00400080	Inc_Loop: ADDI X20, X20, #1
0x00400084	LDUR X0, [SP, #0]
0x00400088	B Loop
0x0040008C	Final_Func: ADDI X0, X19, #0
0x00400090	LDUR LR, [SP, #8]
0x00400094	LDUR X20, [SP, #16]
0x00400098	LDUR X19, [SP, #24]
0x0040009C	ADDI SP, SP, #32
0x004000A0	BR LR
0x004000A4	Check: ADDI X9, XZR, #2
0x004000A8	UDIV X10, X0, X9
0x004000AC	MUL X10, X10, X9
0x004000B0	SUB X10, X0, X10
0x004000B4	CBZ X10, L2
0x004000B8	ADDI X0, XZR, #0
0x004000BC	B Final_ Check
0x004000C0	L2: ADDI X0, XZR, #1
0x004000C4	Final_Check:BR LR

```c
#include <stdio.h>

int is_even(int n) {
    if (n % 2 == 0) {
        return 1;
    } else {
        return 0;
    }
}

int function(int limit) {
    int sum = 0; // X19
```

```
    int i = 0;   // X20

    while (sum < limit) {
        if (is_even(i)) {
            sum = sum + i;
        }
        i++;
    }
    return sum;
}

int main() {
    int result = function(10);
    // printf("Result: %d\n", result); // 12 (0+2+4+6)
    return 0;
}
```

Địa chỉ	Nhãn	Lệnh	Giải thích vai trò
0x00400030	Main:	ADDI X0, XZR, #10	Đặt đối số cho Func là 10. X0 được dùng để truyền đối số đầu tiên.
0x00400034		BL Function	Branch and Link: Gọi hàm Func và lưu địa chỉ lệnh tiếp theo vào LR (X30).
0x00400038		ADD X19, X0, XZR	Lưu giá trị trả về từ Func (nằm trong X0) vào X19.
0x0040003C		...	Các lệnh tiếp theo là để thực hiện lệnh gọi hệ thống (syscall) để kết thúc chương trình.
Hàm Func			
0x00400048	Func:	SUBI SP, SP, #32	Cấp phát 32 byte trên stack frame cho hàm Func.
0x0040004C		STUR ...	Lưu các thanh ghi X19, X20, LR, và đối số X0 lên stack để bảo toàn giá trị.
0x0040005C		ADDI X19, XZR, #0	Khởi tạo biến sum (trong X19) bằng 0.
0x00400060		ADDI X20, XZR, #0	Khởi tạo biến i (trong X20) bằng 0.
0x00400064	Loop:	LDUR X0, [SP, #0]	Tải lại giá trị limit (đối số ban đầu) vào X0 để so sánh.

Địa chỉ	Nhãn	Lệnh	Giải thích vai trò
0x00400064		SUBS XZR, X19, X0	**Điều kiện vòng lặp:** So sánh sum (X19) với limit (X0).
0x00400068		B.LT L1	Nếu sum < limit, nhảy đến L1 để tiếp tục xử lý.
0x0040006C		B FinalFunction	Nếu sum >= limit, thoát khỏi vòng lặp và đi đến phần kết thúc hàm.
0x00400070	L1:	ADDI X0, X20, #0	Chuẩn bị đối số cho hàm Check: X0 = i (X20).
0x00400074		BL Check	Gọi hàm Check(i). Kết quả trả về trong X0 (1 nếu chẵn, 0 nếu lẻ).
0x00400078		CBZ X0, Inc_Loop	Nếu kết quả Check là 0 (i là số lẻ), nhảy tới Inc_Loop.
0x00400080		ADD X19, X19, X20	**Nếu i là số chẵn:** sum = sum + i.
0x0040007C		ADDI X20, X20, #1	**i++** (trong trường hợp i là chẵn).
0x00400088		B Loop	Quay lại đầu vòng lặp.
0x00400084	Inc_Loop:	ADDI X20, X20, #1	**i++** (trong trường hợp i là số lẻ).
0x00400088		B Loop	Quay lại đầu vòng lặp.
0x0040008C	FinalFunc:	ADDI X0, X19, #0	Đặt giá trị trả về của hàm Func là sum (X19).
0x00400090		LDUR ...	Khôi phục các thanh ghi đã lưu từ stack.
0x0040009C		ADDI SP, SP, #32	Giải phóng stack frame.
0x004000A0		BR LR	Trở về hàm Main.
Hàm Check			
0x004000A4	Check:	UDIV/MUL/SUB	Các lệnh này thực hiện phép toán n - (n / 2) * 2, tương đương với n % 2.
0x004000B4		CBZ X10, L2	Nếu kết quả n % 2 bằng 0, nhảy đến L2.

Địa chỉ	Nhãn	Lệnh	Giải thích vai trò
0x004000B8		ADDI X0, XZR, #0	Nếu n lẻ, đặt giá trị trả về là 0.
0x004000C0	L2:	ADDI X0, XZR, #1	Nếu n chẵn, đặt giá trị trả về là 1.
0x004000C4	FinalCheck:	BR LR	Trở về hàm Func.