

Final Review

ICS312 - Spring 2009 Machine-Level and Systems Programming

Henri Casanova (henric@hawaii.edu)

What to Study for the Final?

- Comprehensive and open note
 - No Virtual Memory
 - No Syntactic Analysis
- Types of questions
 - No quiz-like questions
 - Only problem-like questions
 - Write a piece of code to do something
- What to study
 - All lecture notes
 - Homework assignment answers
 - Midterm answers

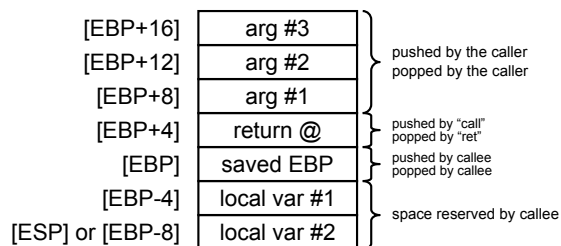
“Obvious” Possible Questions

- Data segments
- Carry and Overflow bits
- Write a function with arguments and parameters
- Translate some C code into assembly
- Array addressing
- Write a piece of code that uses/creates bit patterns
- Conversions
 - 2's-complement
 - FP values
- Write a piece of code that does FP arithmetic
- Draw a picture of the stack at some point in a program
- Convert some NFAs to REs

“Obvious” Possible Questions

- Data segments
- Carry and Overflow bits
- Write a function with arguments and parameters
- Translate some C code into assembly
- Array addressing
- Write a piece of code that uses/creates bit patterns
- Conversions
 - 2's-complement
 - FP values
- Write a piece of code that does FP arithmetic
- Draw a picture of the stack at some point in a program
- Convert some NFAs to REs

The Main Stack Structure



Write a Function that ...

- Write a function that takes 2 addresses as arguments and swaps their content, without destroying any register
 - watch out for indirections...

Write a Function that ...

- Write a function that takes 2 addresses as arguments and swaps their content, without destroying any register

```
f:  push  ebp
    mov  ebp, esp
    pusha
    mov  eax, [ebp+8]    ; eax = 1st address
    mov  ebx, [eax]     ; ebx = content at 1st address
    mov  ecx, [ebp+12]  ; ecx = 2nd address
    mov  edx, [ecx]     ; edx = content at 2nd address
    mov  [eax], edx     ; store edx to 1st address
    mov  [ecx], ebx     ; store ebx to 2nd address
    popa
    pop  ebp
    ret
```

Translate this Function...

```
float f(int *x) {
    int y;
    y = *x+1;
    return (float)y;
}
```

- The return value is FP (so not returned in eax)
- Let's not destroy any register
- Let's use the local variable in our translation

Translate this Function...

```
float f(int *x) {
    int y;
    y = *x+1;
    return (float)y;
}
```

```
f:  push  ebp
    mov  ebp, esp
    sub  esp, 4          ; reserve space for y
    push ebx           ; save ebx
    mov  ebx, [ebp+8]   ; set ebx to x
    mov  ebx, [ebx]     ; set ebx to *x
    inc  ebx           ; set ebx to *x + 1
    mov  [ebp-4], ebx   ; set y to *x + 1
    fild dword [ebp-4] ; put (float)y on top of the FP stack
    pop  ebx           ; restore ebx
    add  esp, 4        ; remove space for y
    pop  ebp
    ret
```

Array Addressing

- Consider the following declaration: `int y[17][12];`
- Assume that the 2-byte address of `y[0,0]` is `0240h`
- What's the address of `y[10,3]`?

Array Addressing

- Consider the following C declaration: `int y[17][12];`
- Assume that the 2-byte address of `y[0,0]` is `0240h`
- What's the hex address of `y[10,3]`?

$$\begin{aligned} @ &= 0240h + 12 * 10 * 4 + 3 * 4 &&= 0240h + 492d \\ & &&= 0240h + 01ECh \\ & &&= 042Ch \end{aligned}$$

Bit Patterns...

- Write a piece of code that flips the 5-th bit (rightmost bit is bit #0) in `ebx`

Bit Patterns...

- Write a piece of code that flips the 5-th bit (rightmost bit is bit #0) in ebx

```
xor ebx, 020h
```

Bit Patterns...

- Write a piece of code that turns off the n-th bit (rightmost bit is bit #0) in ebx, where n is stored in al

Bit Patterns...

- Write a piece of code that turns off the n-th bit (rightmost bit is bit #0) in ebx, where n (<31) is stored in al

```
mov cl, al
mov eax, 01h
shl eax, cl
not eax
and ebx, eax
```

FP Value conversion

- Assume a machine that stores 8-bit FP values as a 5-bit number and a 3-bit exponent as 1.nnnnnee. What is the binary representation of 7.42?

FP Value conversion

- Assume a machine that stores 8-bit FP values as a 5-bit number (1.nnnnn) and a 3-bit exponent (eee): nnnnnee. What is the binary representation of 7.42?
 - Converting 7 to binary:
 - $7d = 111b$
 - Converting .42 to binary:
 - $.42 * 2 = 0.84$
 - $.84 * 2 = 1.68$
 - $.68 * 2 = 1.36$
 - $.36 * 2 = 0.72$
 - Therefore: $7.42d = 111.0110b = 1.110110b * 2d^{(010b)}$
 - Answer: 11011 010

FP Arithmetic..

- Write a function that takes two float arguments, x and y, and returns $1+x*y$

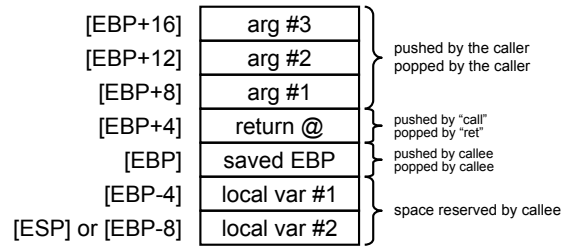
FP Arithmetic..

- Write a function that takes two float arguments, x and y, and returns 1+x*y

```
f:  push  ebp
    mov   ebp, esp
    fld   [ebp+8]      ; st0 = x
    fld   [ebp+12]     ; st0 = y, st1 = x
    fmul  st1          ; st0 = x*y, st1 = y
    ffree st1          ; st0 = x*y
    fld1  [ebp+4]      ; st0 = 1, st1 = x*y
    fadd  st1          ; st0 = 1+x*y, st1 = x*y
    ffree st1          ; st0 = 1+x*y
    (mov  esp, ebp)
    pop   ebp
    ret
```

Draw the Stack...

Any questions about the midterm/homework?
Should we do one of those again?



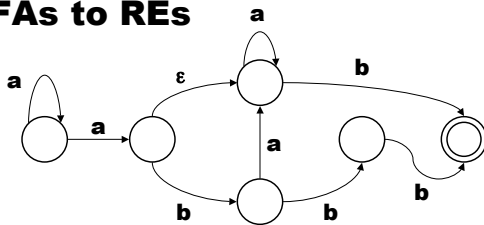
Draw the Stack...

```
...          g(int x, y) {
f(3,4);      int z;
...          z = z+y;
            f(x, z);
f(int x, int y) {
if (x > 0)   }
    g(x-1, y+1);
else
    print ("Here\n");
}
```

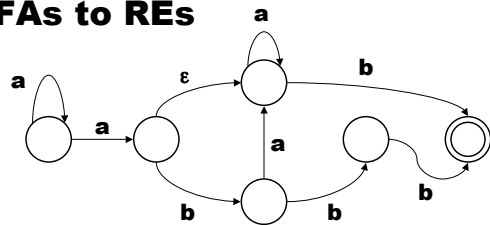
Draw the Stack...

```
f(3,4);
f(int x, int y) {
if (x > 0)
    g(x-1, y+1);
else
    print ("Here\n");
}
g(int x, y) {
int z;
z = z+y;
f(x, z);
}
4
3
ret @
saved ebp
5
2
ret @
saved ebp
z=7
7
2
ret @
saved ebp
8
1
ret @
saved ebp
z=9
9
1
ret @
saved ebp
10
0
ret @
saved ebp
z=10
10
0
ret @
saved ebp
```

NFAs to REs



NFAs to REs



$$\begin{aligned}
 \text{RE} &= a^+a^*b \mid a^+ba^*b \mid a^+bbb \\
 &= a^+b \mid a^+ba^*b \mid a^+bbb \\
 &= a^+b(\epsilon \mid a^*b \mid bb)
 \end{aligned}$$