Addresses and Pointers

A pointer is a data object that holds an address.

The value 240 is stored at address 0x20.

A pointer stored at address 0x08 points to address 0x20.

A pointer to a pointer is stored at address 0x00.

Control flow indirection

First, two more quick notes about loops and arrays, then:

Return of the JR: Switch and Dynamic Dispatch

1. While → Do-While

```c
void copy(int src[], int dest[], int n) {
  int i = 0;
  while (i < n) {
    dest[i] = src[i];
    i++;
  }
}
```

Compiler transforms to do-while form. Why?

```c
void copy(int src[], int dest[], int n) {
  int i = 0;
  goto start;
  do {
    dest[i] = src[i];
    i++;
  } while (i < n);
  start:
}
```

1. While → Do-While

```c
loop:
    bge $t0, $a3, loop_end
...} loop
loop_end:
...
```

Compiler transforms to do-while form. Why?

```c
loop:
...} loop_test
loop:
...} loop_test:
    bhi $t0, $a3, loop
...} loop
```
2. array indexing → pointer arithmetic

void copy(int[] src, int[] dest, int n) {
  int i = 0;
  while (i < n) {
    dest[i] = src[i];
    i++;
  }
}

Jt

loop:
li $t2, 0
j loop_test
loop:
add $t0, $a0, $t4
lw $t3, 0($t0)
add $t1, $a0, $t4
sw $t3, 0($t1)
addi $t2, $t2, 4
loop_test:
blt $t2, $a2, loop

Switch Statement

int x = ...;
int y = ...;
int w = 1;
switch(x) {
  case 1:
    w = y*z;
    break;
  case 2:
    w = y/z;
    /* Fall Through */
  case 3:
    w += z;
    break;
  case 5:
  case 6:
    w -= z;
    break;
  default:
    w = 2;
}

Jump Table Structure

Switch Statement

switch(x) {
  case val_0:
    Block 0
  case val_1:
    Block 1
  . .
  case val_n:
    Block n
}

Jump Table

Approximate Translation

if (x <= 6)
  target = jt[x];
  goto target;
switch_end:
Jump Table

Switch Statement

Cases (Partial)

Would you implement these with jump tables?
On translation, layout, and implementation

We show natural, common, or conventional translations.

Java: No guarantee of this implementation/layout. Language is (mostly clean) abstraction.

C: Much of implementation/layout guaranteed. Language exposes many machine details.

Dynamic Dispatch

```
public class Shape { ...  
  public double area(); { ... }  
}  
public class Rectangle { ...  
  public double area(); { ... }  
}  
public class Triangle { ...  
  public double area(); { ... }  
}  
...
```

```
public void printArea(Shape s) {  
  System.out.println("Share area is: "+ s.area());  
}  
...
```

```
printArea(new Rectangle(...));  
printArea(new Triangle(...));
```

Implementing Dynamic Dispatch

```
  a - 12  
  a - 8  
  a - 4  
  v  

Rectangle vtable:  
  equals(Object o)  
  clone()  
  toString()  
  area()  

v + 4  

a - 12  
  v  

v + 8  

a - 12  
  v  

v + 12  

v + 16
```

```
null check? Add field, methods in subclass?  
Why is multiple inheritance a problem?
```

```
Which area() gets called?
```