根据你提供的PDF文档内容，我们可以将给定的Rust代码翻译成LLVM IR格式。以下是给定的Rust代码：

```
let a[10]:int = {1,2,3,4,5,6,7,8,9,10};
fn binsearch(x:int) -> int {
    let high:int = 9;
    let low:int = 0;
    let mid:int = (high+low)/2;
    while(a[mid]!=x && low < high) {
        mid=(high+low)/2;
        if(x<a[mid]) {
            high = mid-1;
        } else {
            low = mid +1;
        }
    }
    if(x == a[mid]) {
        ret mid;
    }
    else {
        ret -1;
    }
}

fn main() -> int {
    let r = binsearch(2);
    ret r;
}
```

转换为LLVM IR

根据PDF中的例子和格式，下面是转换后的LLVM IR代码：

```
@a = global [10 x i32] [i32 1, i32 2, i32 3, i32 4, i32 5, i32 6, i32 7, i32 8, i32 9,

define i32 @binsearch(i32 %x) {
entry:
    %high = alloca i32
    %low = alloca i32
    %mid = alloca i32
    %x.addr = alloca i32
    store i32 %x, i32* %x.addr
    store i32 9, i32* %high
    store i32 0, i32* %low

    br label %cond

cond:
    %0 = load i32, i32* %low
    %1 = load i32, i32* %high
    %2 = add i32 %0, %1
    %midval = sdiv i32 %2, 2
    store i32 %midval, i32* %mid
    %3 = load i32, i32* %mid
    %4 = getelementptr [10 x i32], [10 x i32]* @a, i32 0, i32 %3
    %5 = load i32, i32* %4
    %6 = load i32, i32* %x.addr
    %cmp1 = icmp eq i32 %5, %6
    %7 = load i32, i32* %low
    %8 = load i32, i32* %high
    %cmp2 = icmp slt i32 %7, %8
    %cond = and i1 %cmp1, %cmp2
    br i1 %cond, label %body, label %end

body:
    %9 = load i32, i32* %x.addr
    %10 = load i32, i32* %mid
    %11 = getelementptr [10 x i32], [10 x i32]* @a, i32 0, i32 %10
    %12 = load i32, i32* %11
    %cmp3 = icmp slt i32 %9, %12
    br i1 %cmp3, label %if.then, label %if.else

if.then:
    %13 = load i32, i32* %mid
    %14 = sub i32 %13, 1
    store i32 %14, i32* %high
```

```
    br label %cond

if.else:
    %15 = load i32, i32* %mid
    %16 = add i32 %15, 1
    store i32 %16, i32* %low
    br label %cond

end:
    %17 = load i32, i32* %mid
    %18 = getelementptr [10 x i32], [10 x i32]* @a, i32 0, i32 %17
    %19 = load i32, i32* %18
    %20 = load i32, i32* %x.addr
    %cmp4 = icmp eq i32 %19, %20
    br i1 %cmp4, label %found, label %notfound

found:
    %21 = load i32, i32* %mid
    ret i32 %21

notfound:
    ret i32 -1
}

define i32 @main() {
entry:
    %r = call i32 @binsearch(i32 2)
    ret i32 %r
}
```

解释

- **全局数组**：使用 `@a` 声明并初始化全局数组。
- **`binsearch` 函数**：
  - 在入口块分配 `high`，`low`，`mid`，和 `x.addr`。
  - 初始化 `high` 和 `low` 的值。
  - 使用 `br` 指令进行条件分支和循环。
  - 在 `cond` 块中，计算 `mid` 的值并进行比较。
  - 根据比较结果，在 `body` 块中调整 `high` 和 `low` 的值。
  - 在 `end` 块中，检查最终结果，决定返回 `mid` 或 `-1`。
- **`main` 函数**：

- 调用 `binsearch` 函数并返回结果。

这种转换遵循LLVM IR的结构和指令集，能够体现原始Rust代码的逻辑和控制流。