

```
entry:  
%A = alloca [10 x i32], align 16  
%B = alloca [10 x i32], align 16  
call void @llvm.memcpy.p0.p0.i64(ptr align 16 %A, ptr align 16  
... @_const.main.A, i64 40, i1 false)  
call void @llvm.memset.p0.i64(ptr align 16 %B, i8 0, i64 40, i1 false)  
br label %for.cond
```

```
for.cond:  
%i.0 = phi i32 [ 0, %entry ], [ %inc, %for.inc ]  
%j.0 = phi i32 [ 0, %entry ], [ %j.1, %for.inc ]  
%cmp = icmp slt i32 %i.0, 10  
br i1 %cmp, label %for.body, label %for.end
```

T	F
---	---

```
for.body:  
%idxprom = sext i32 %j.0 to i64  
%arrayidx = getelementptr inbounds [10 x i32], ptr %A, i64 0, i64 %idxprom  
%0 = load i32, ptr %arrayidx, align 4  
%mul = mul nsw i32 %0, 13  
%add = add nsw i32 %mul, 4  
%add1 = add nsw i32 %add, %i.0  
%idxprom2 = sext i32 %i.0 to i64  
%arrayidx3 = getelementptr inbounds [10 x i32], ptr %B, i64 0, i64 %idxprom2  
store i32 %add1, ptr %arrayidx3, align 4  
%rem = srem i32 %i.0, 8  
%cmp4 = icmp eq i32 %rem, 0  
br i1 %cmp4, label %if.then, label %if.end
```

T	F
---	---

```
if.then:  
br label %if.end
```

```
if.end:  
%j.1 = phi i32 [ %i.0, %if.then ], [ %j.0, %for.body ]  
%idxprom5 = sext i32 %i.0 to i64  
%arrayidx6 = getelementptr inbounds [10 x i32], ptr %B, i64 0, i64 %idxprom5  
%1 = load i32, ptr %arrayidx6, align 4  
%call = call i32 (ptr, ...) @printf(ptr noundef @.str, i32 noundef %1)  
br label %for.inc
```

```
for.inc:  
%inc = add nsw i32 %i.0, 1  
br label %for.cond, !llvm.loop !6
```

```
for.end:  
ret i32 0
```

CFG for 'main' function