



東邦大学

いのち
生命の科学で未来をつなぐ

ループを書く



ループ ⇒ for 文だ



東邦大学

ループ \Rightarrow for 文だ
for (i=0; i<10; i++){
 x = x+i;
}

ループ \Rightarrow for 文だ
for (i=0; i<10; i++){
 x = x+i;
}

i が 1 から 10 までの間、 $x=x+i$



フローチャート (流れ図)

```
for (i=0; i<10; i++) x=x+i;
```

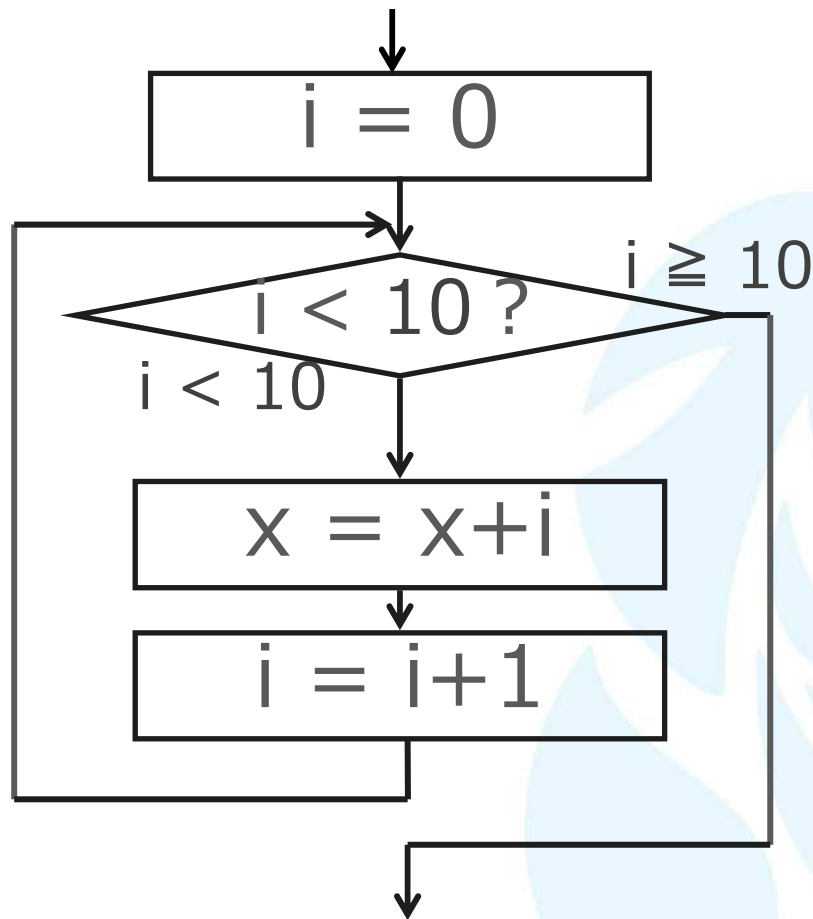
フローチャート（流れ図）

```
for (i=0; i<10; i++) x=x+i;
```

どこかで習いましたか？

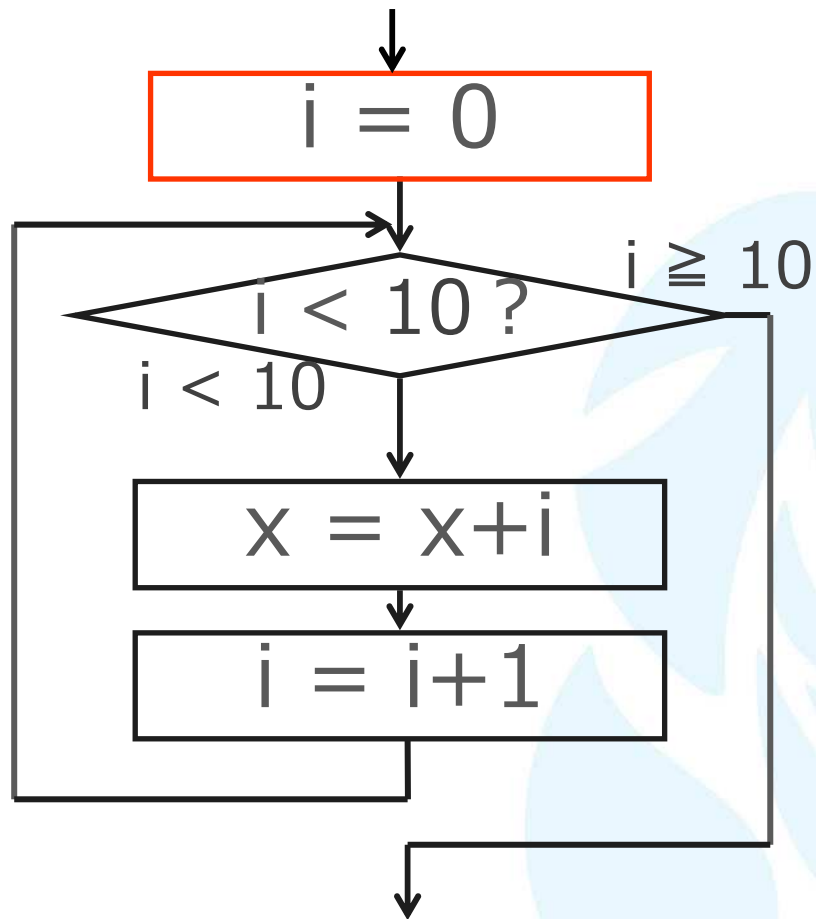
フローチャート (流れ図)

```
for (i=0; i<10; i++) x=x+i;
```



フローチャート (流れ図)

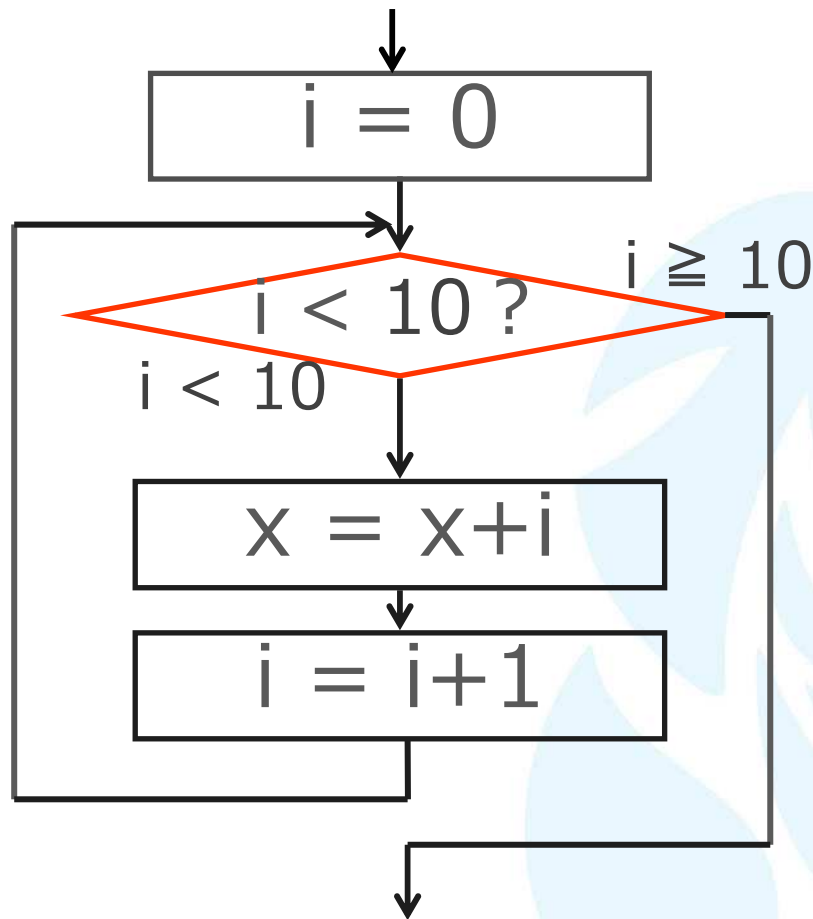
```
for (i=0; i<10; i++) x=x+i;
```



i を初期値 0 にする

フローチャート (流れ図)

```
for (i=0; i<10; i++) x=x+i;
```

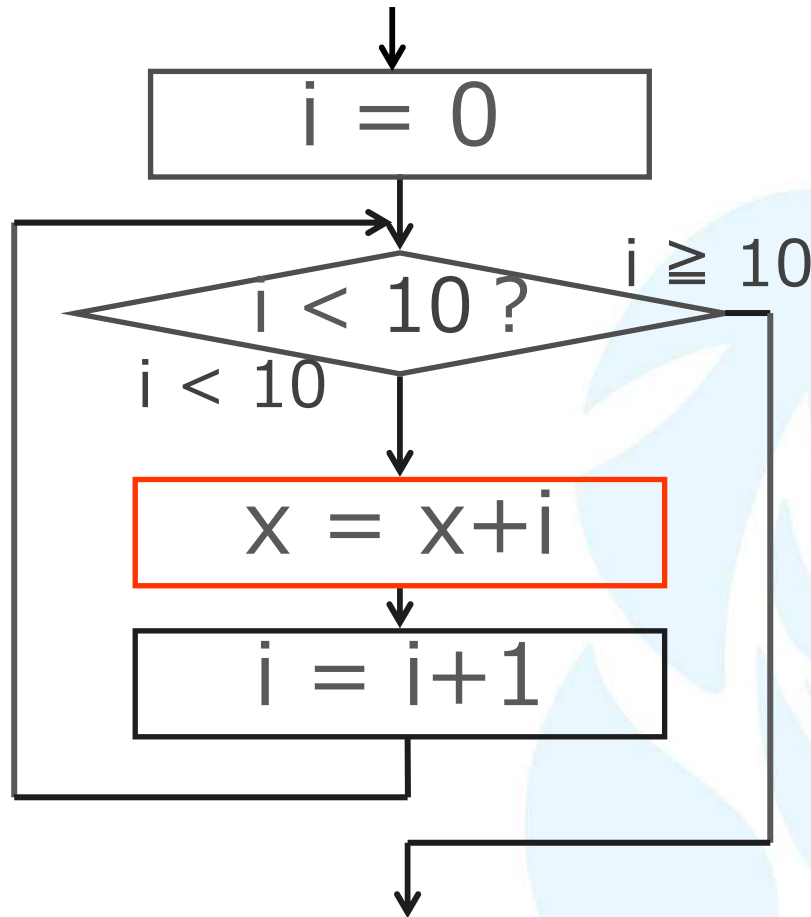


i を初期値 0 にする

i < 10 の間 繰り返す
i ≥ 10 なら脱出

フローチャート (流れ図)

```
for (i=0; i<10; i++) x=x+i;
```



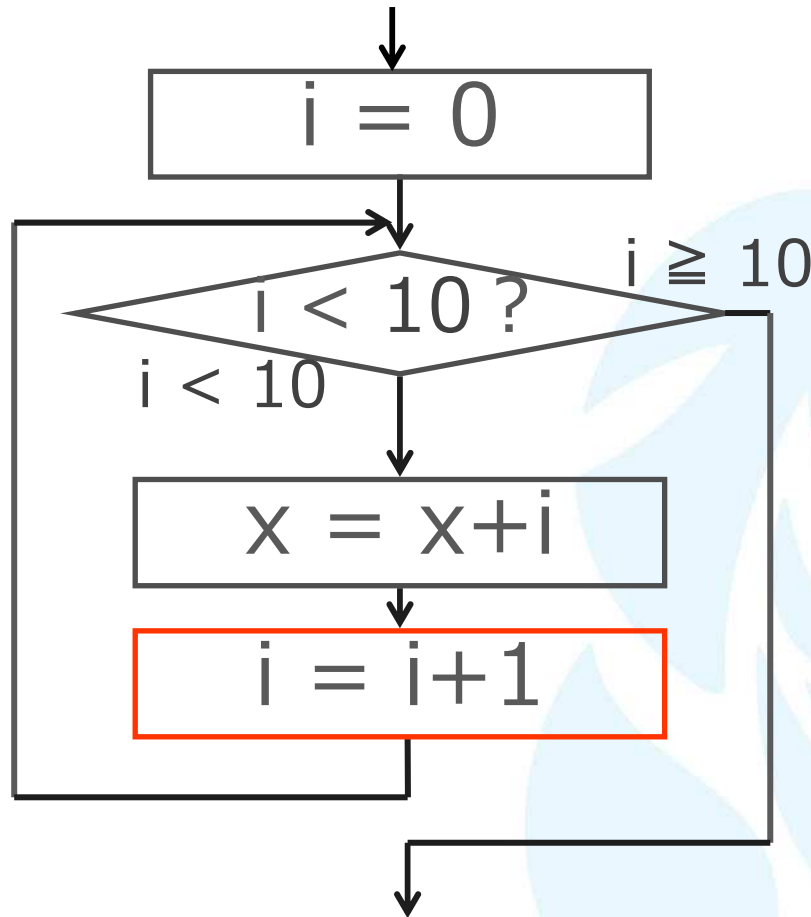
i を初期値 0 にする

i < 10 の間 繰り返す
i ≥ 10 なら脱出

x = x + i (ループ本体)

フローチャート (流れ図)

```
for (i=0; i<10; i++) x=x+i;
```



i を初期値 0 にする

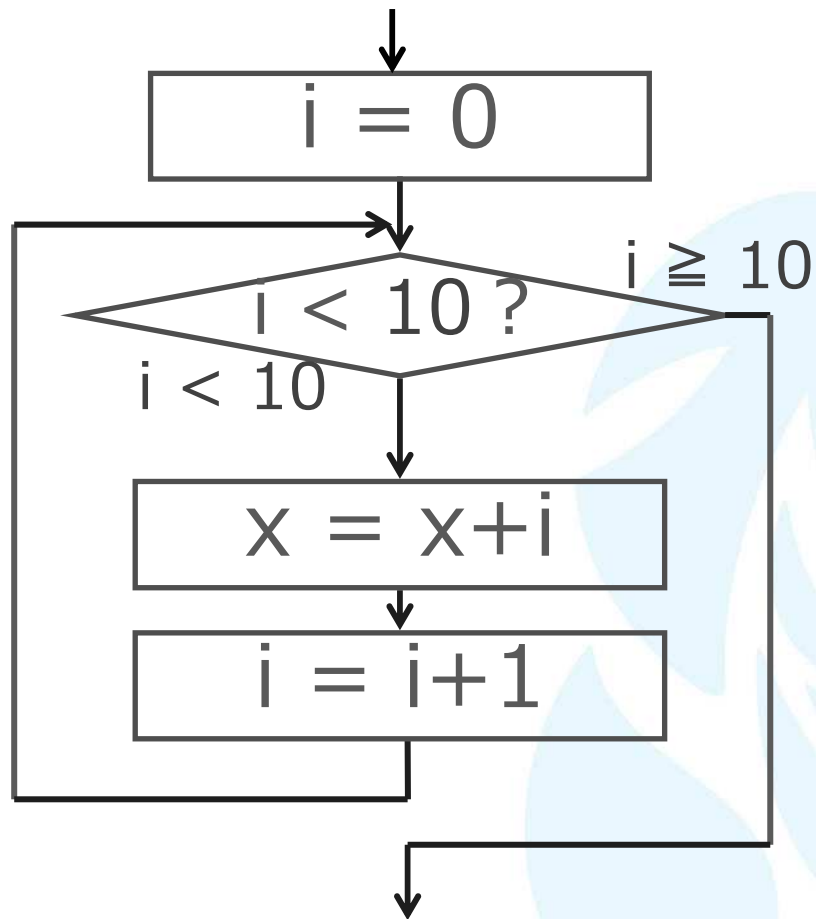
i < 10 の間 繰り返す
i ≥ 10 なら脱出

x = x + i (ループ本体)

i = i + 1 (i++) をして
先頭へ戻る

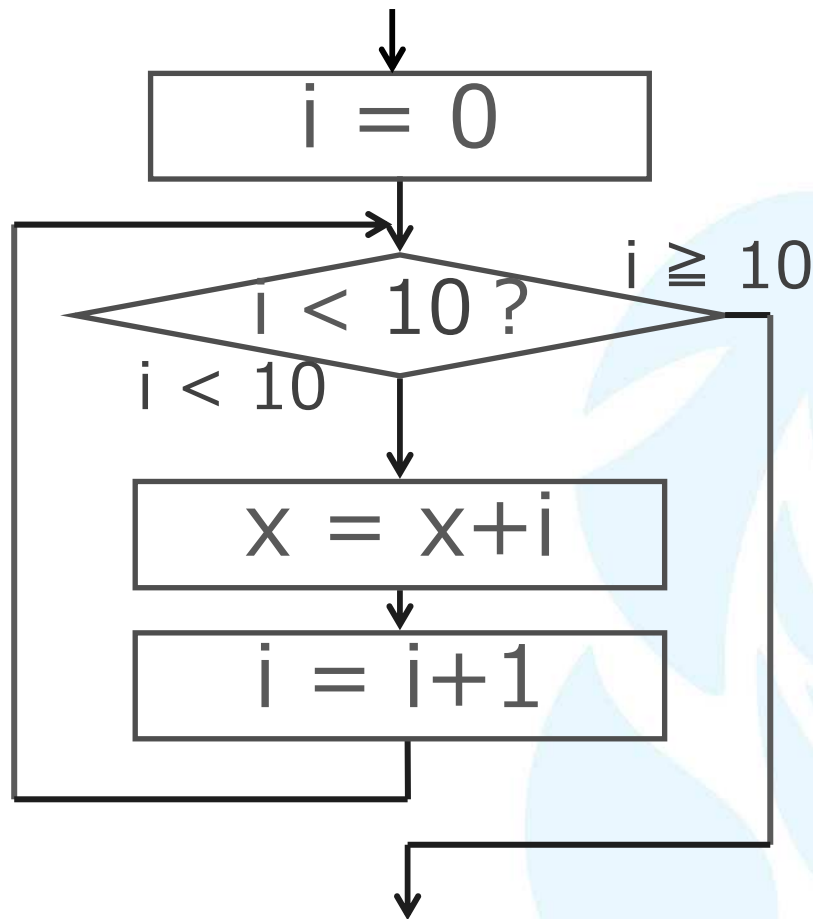
機械命令に変換すると

全体に1列だけなので、そのまま変換できる



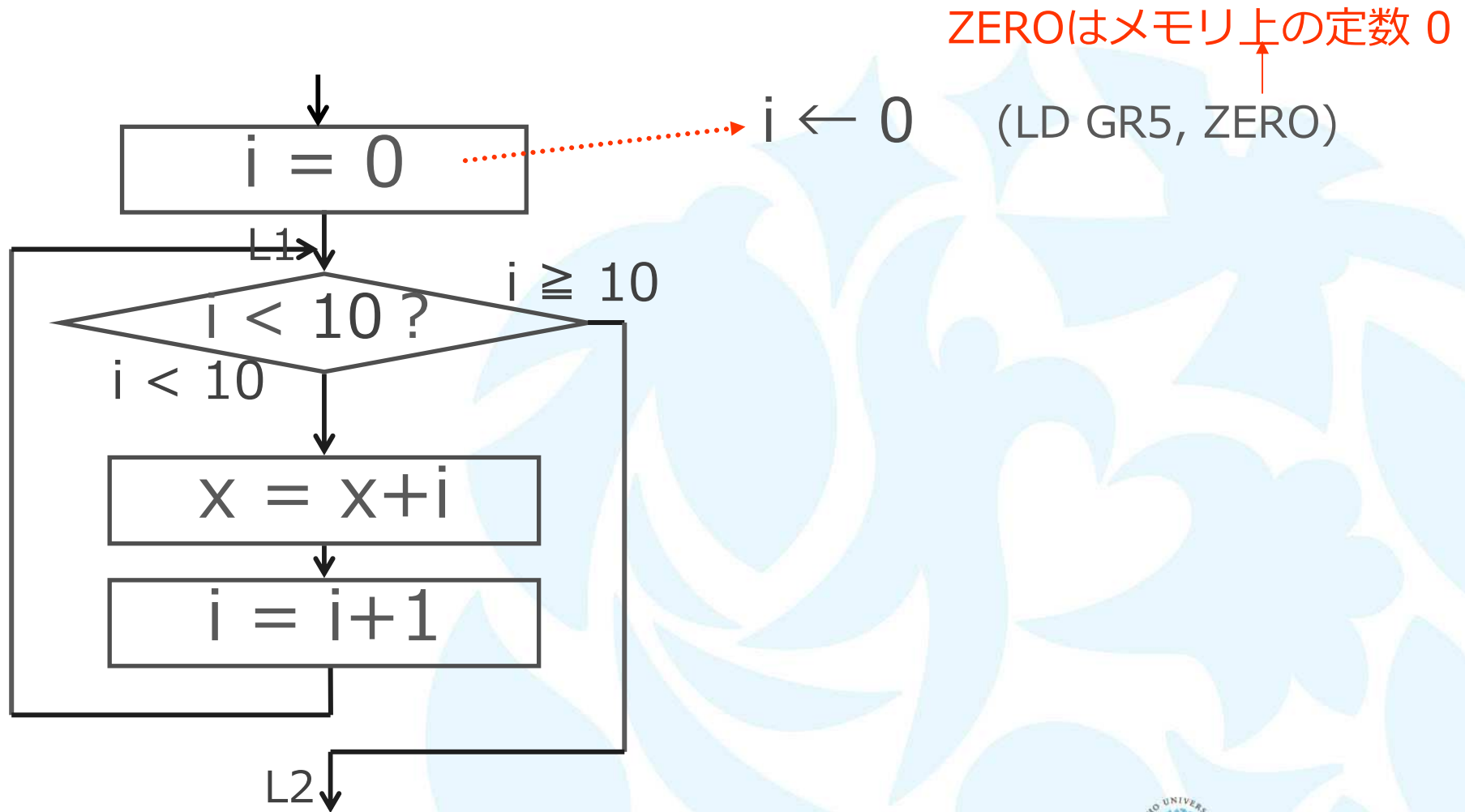
機械命令に変換すると

全体に1列だけなので、そのまま変換できる

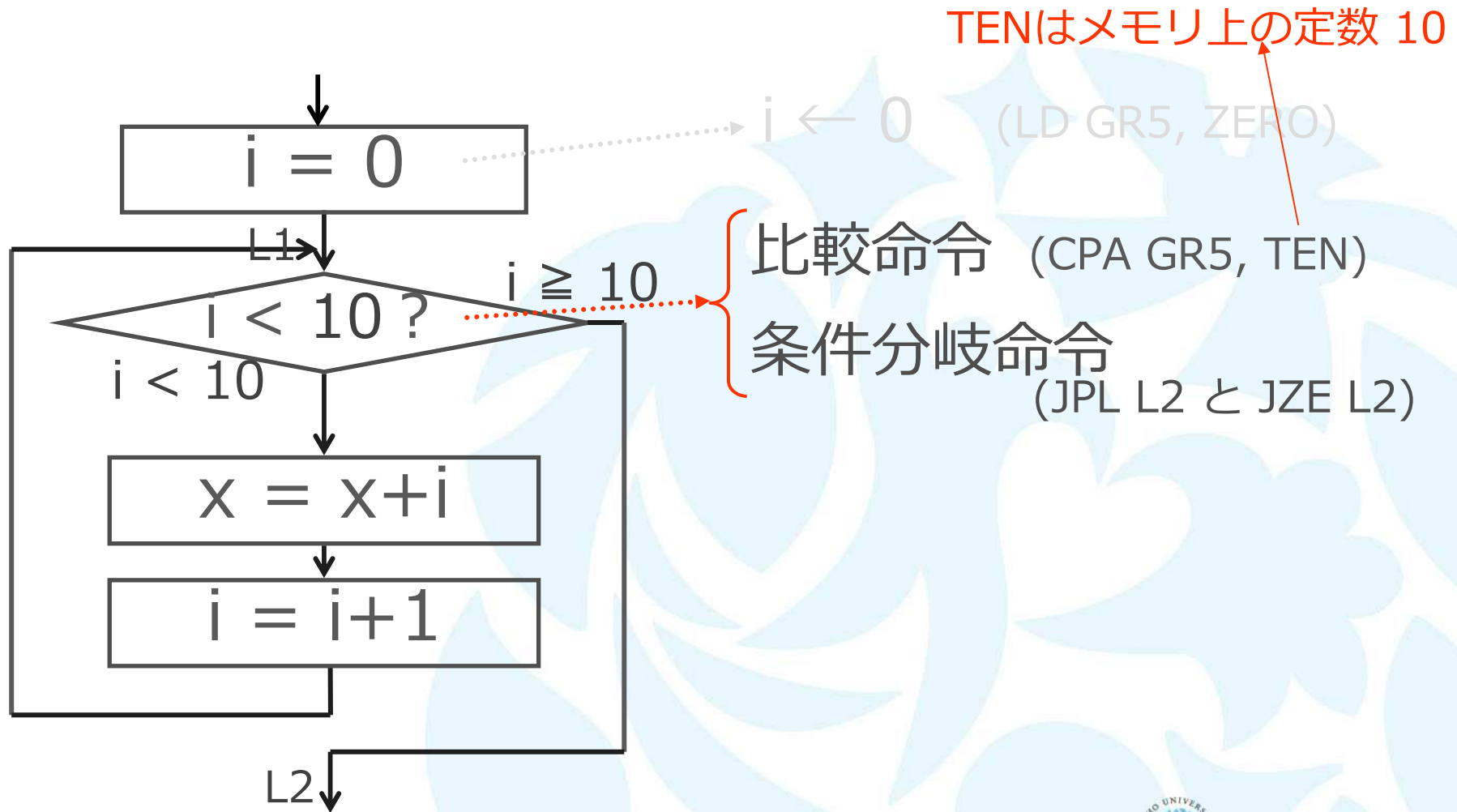


変数 i は
メモリに置かず
レジスタ上にだけ
持つことにする

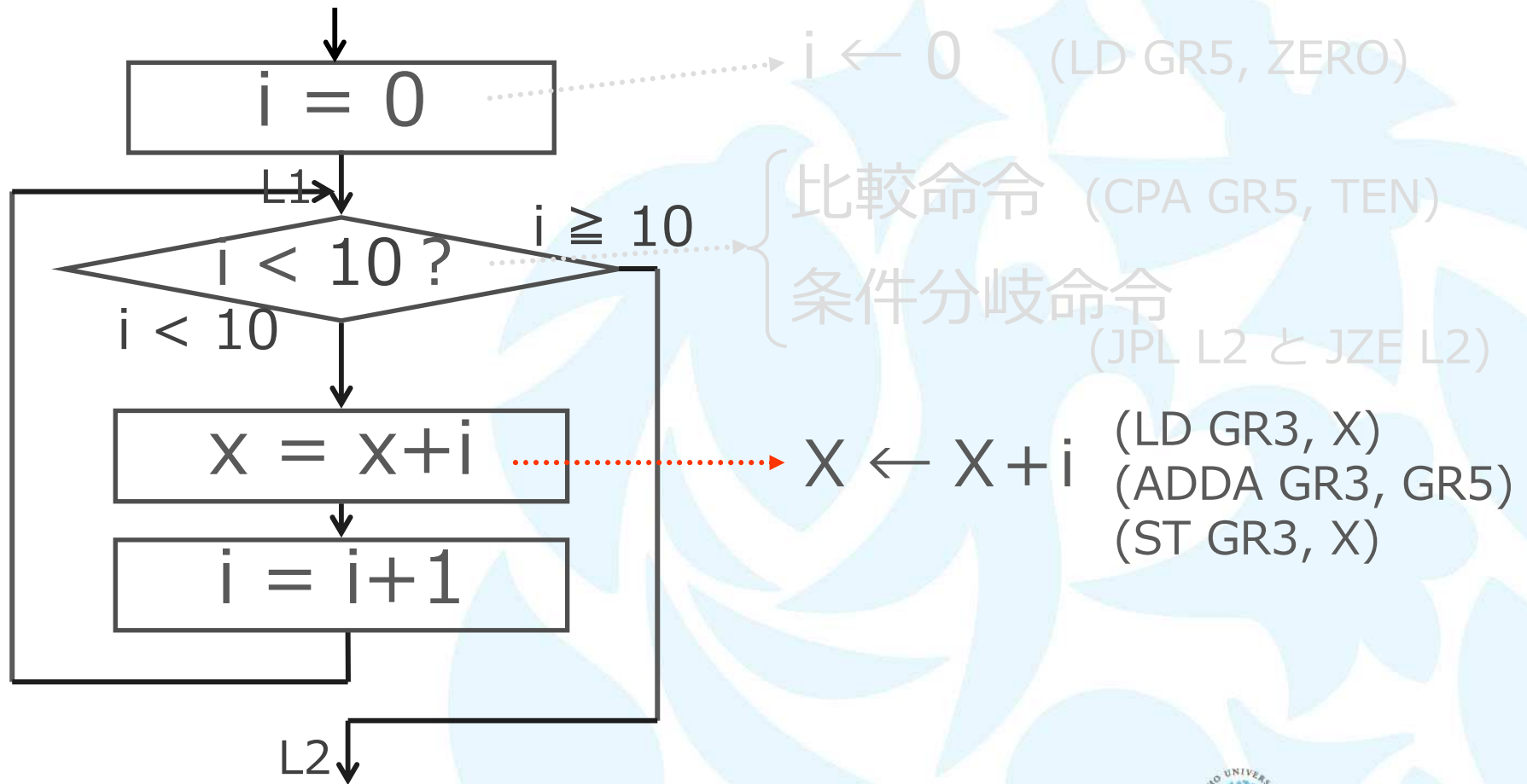
機械命令に変換すると



機械命令に変換すると

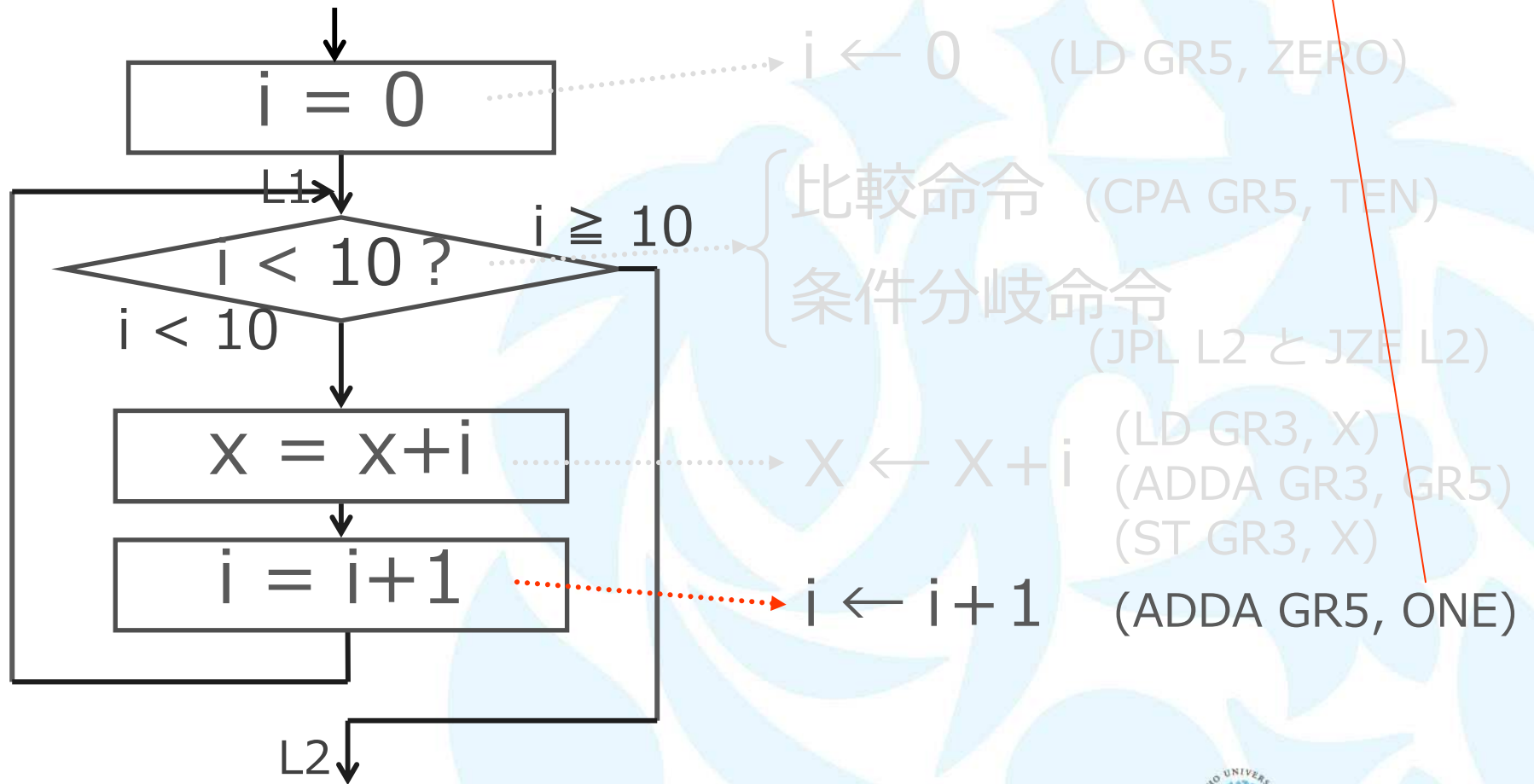


機械命令に変換すると

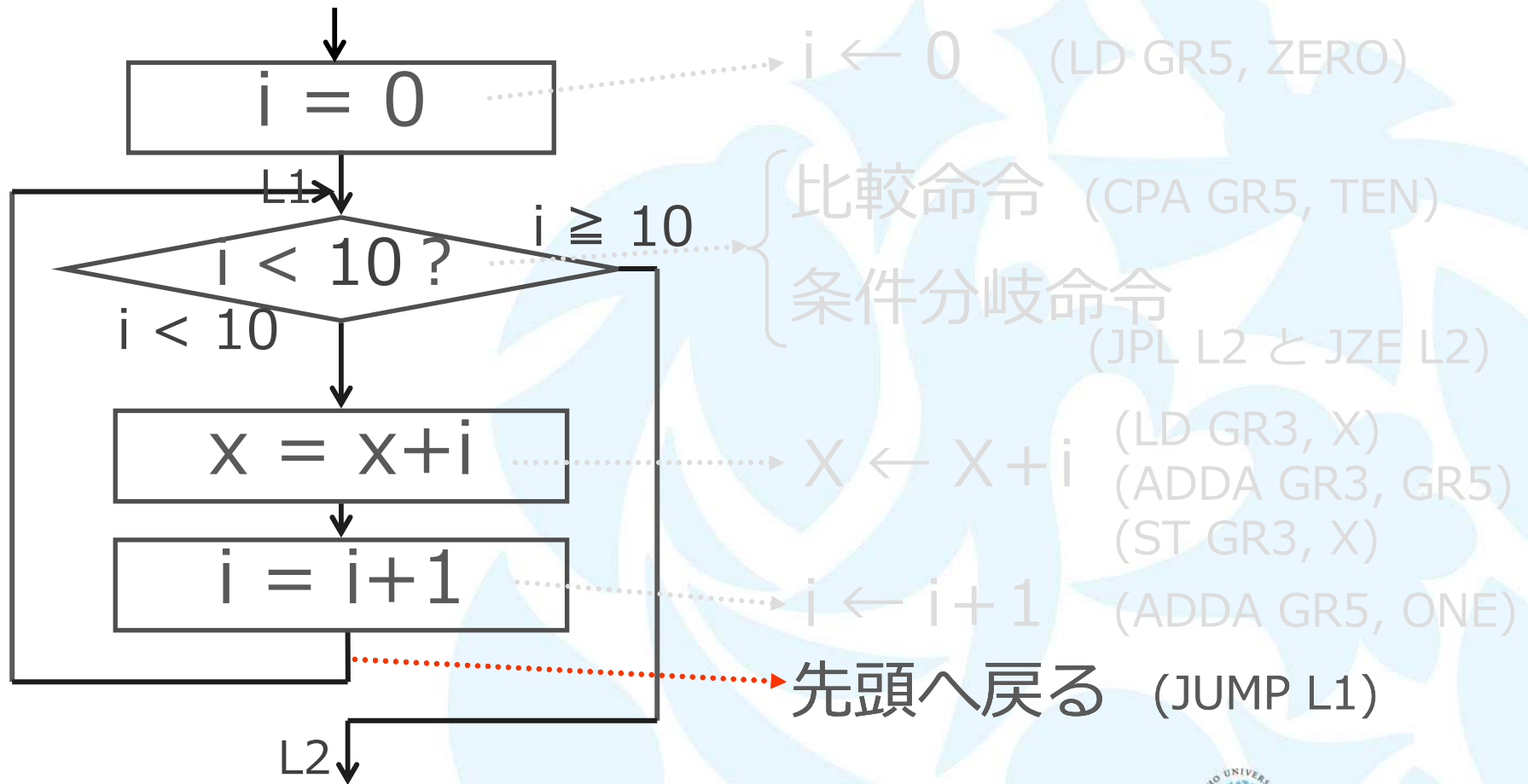


機械命令に変換すると

ONEはメモリ上の定数 1

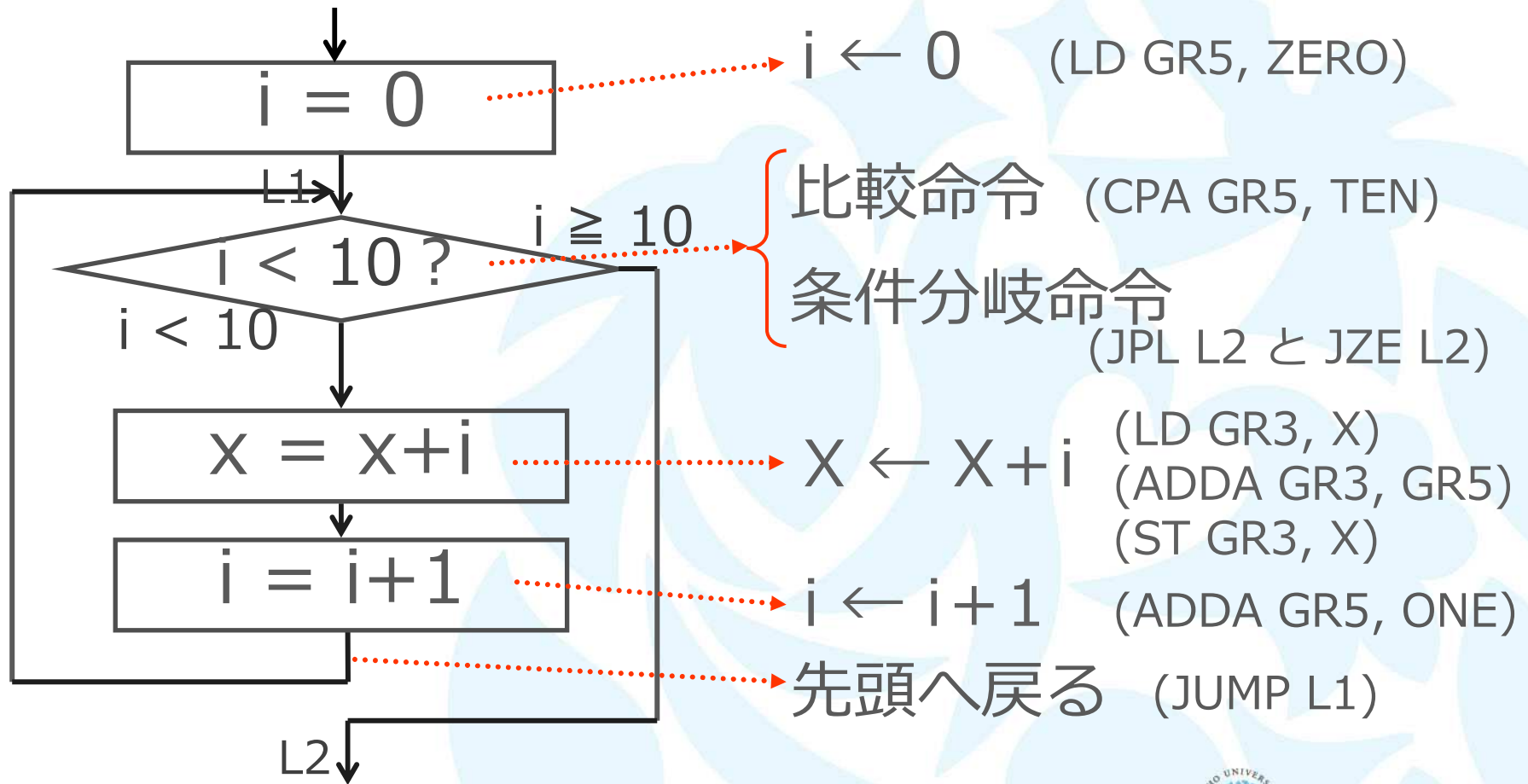


機械命令に変換すると



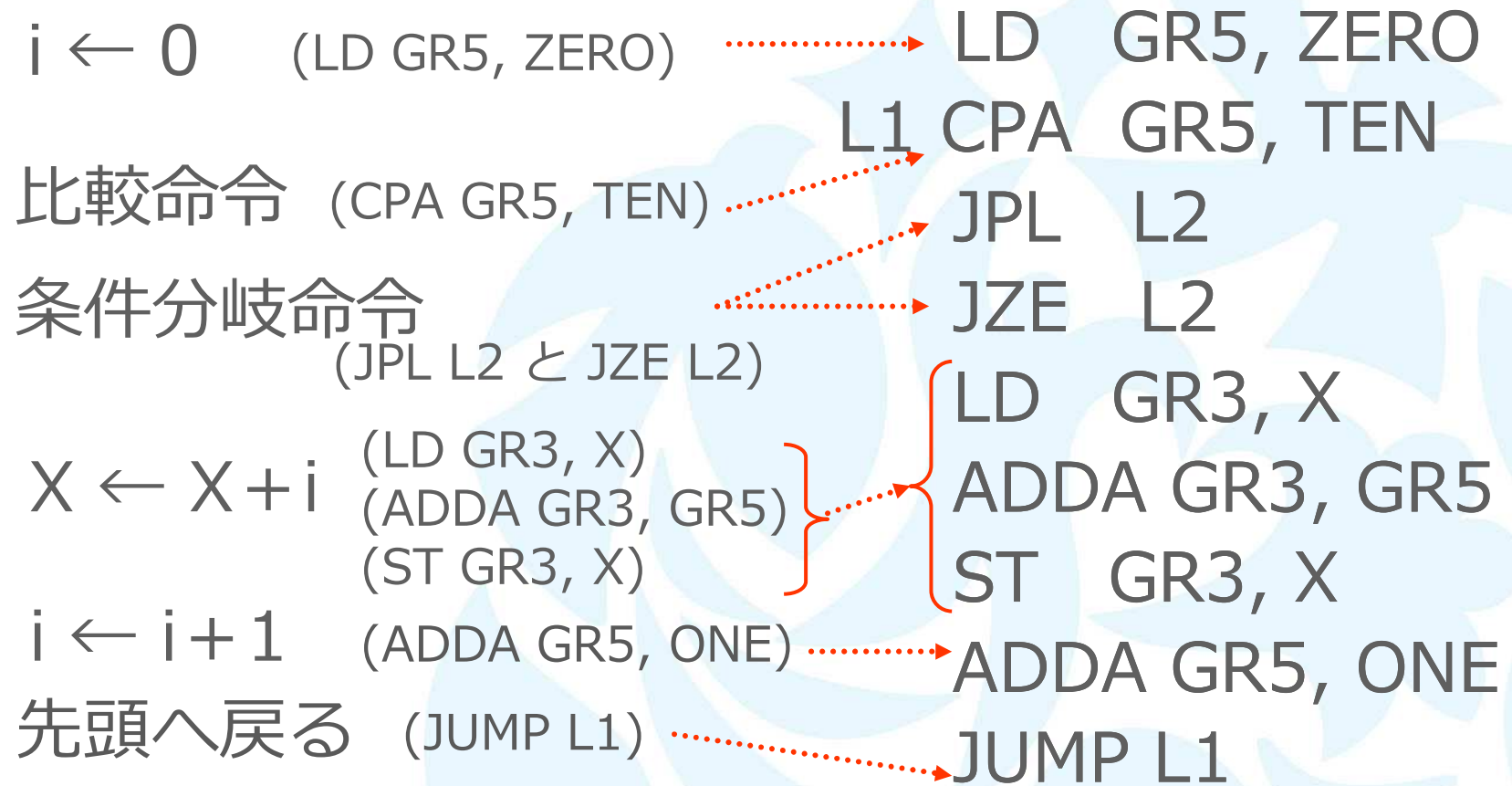
機械命令に変換すると

全部合わせると

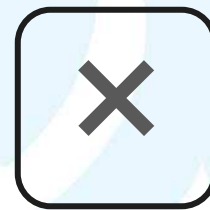
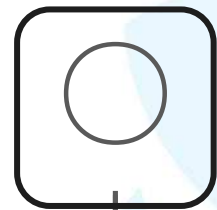


機械命令に変換すると

全部合わせると



for ループと機械命令の関係が
わかりましたか？



↓
次へ

では、演習問題で試してみましよう

演習問題

1 から 100 までの和を求めよ

```
S = 0;
```

```
for (i=1; i≤100; i++) S=S+i;
```



演習問題

```
S = 0;  
for (i=1; i≤100; i++) S=S+i;
```

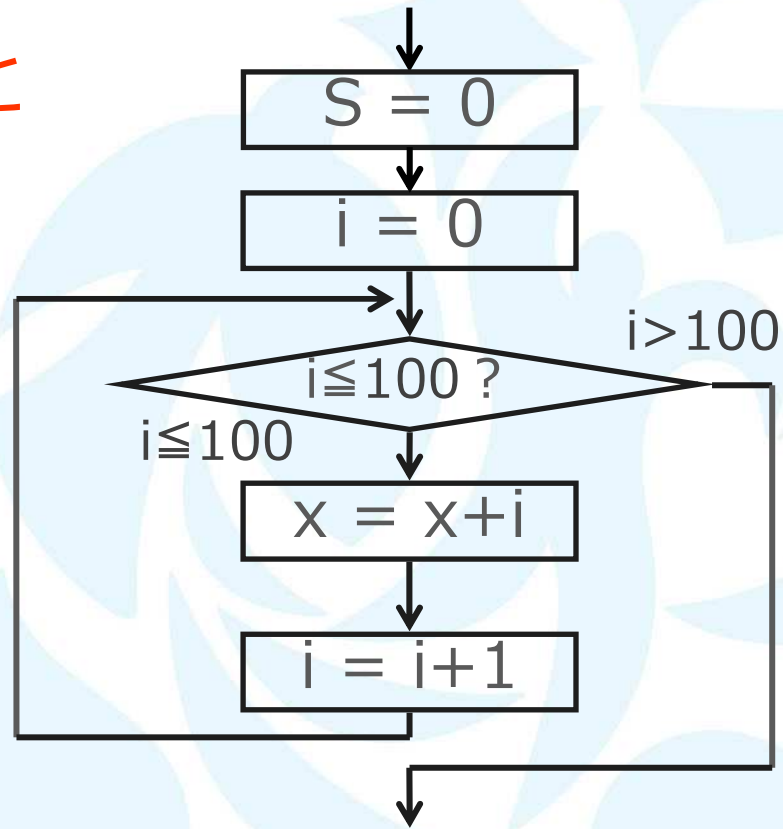
流れ図を描くと



演習問題

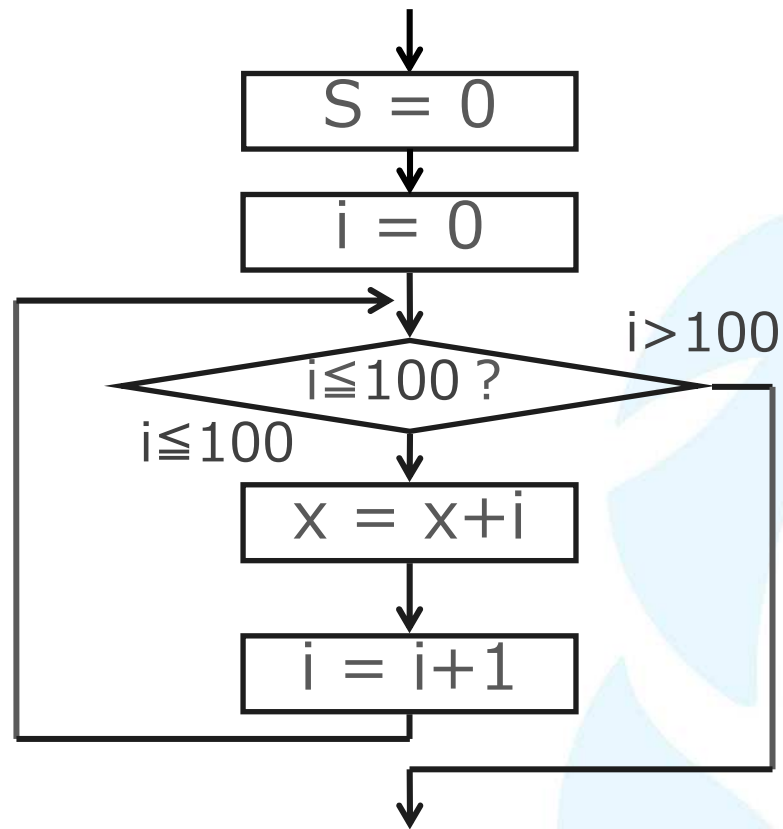
```
S = 0;  
for (i=1; i≤100; i++) S=S+i;
```

流れ図を描くと



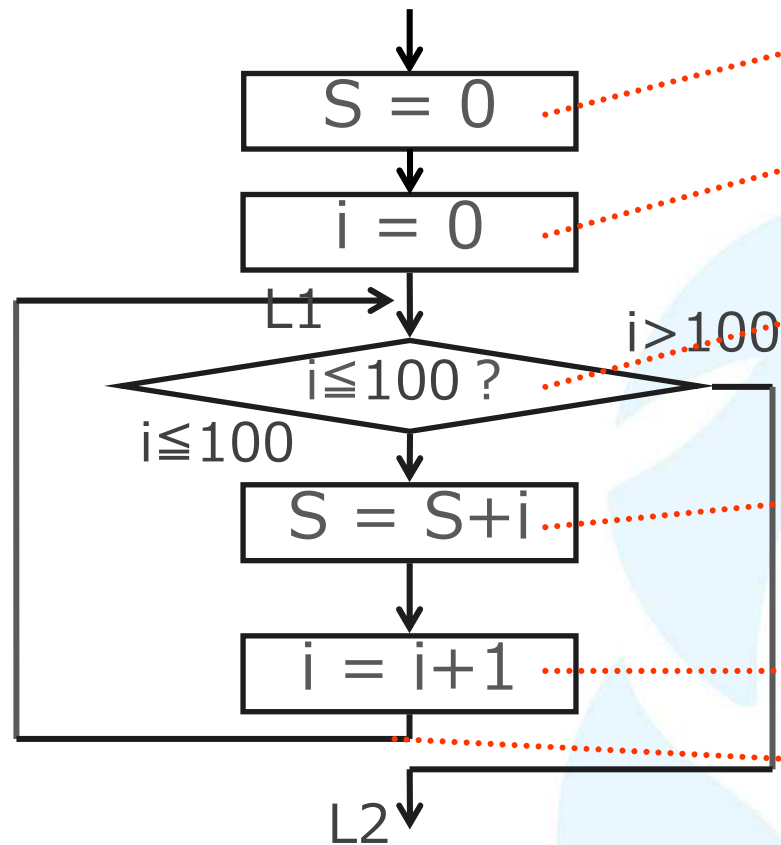
演習問題

あとは自分で出来ますね



演習問題

プログラム例



```
LD GR3, ZERO  
ST GR3, S  
LD GR5, ZERO (GR5 ← 0)
```

```
L1 CPA GR5, HYAKU (i - 100)  
JPL L2 (正 → L2)
```

```
LD GR3, S  
ADDA GR3, GR5 (S = S + i)  
ST GR3, S
```

```
ADDA GR5, ONE  
JUMP L1
```

L2 次の命令

i をGR5に置くことにする

実はもう少し工夫が出来る

```
LD GR3, ZERO
ST GR3, S
LD GR5, ZERO (GR5←0)
```

```
L1 CPA GR5, HYAKU (i-100)
JPL L2 (正→L2)
```

```
LD GR3, S
ADDA GR3, GR5 (S=S+i)
ST GR3, S
```

```
ADDA GR5, ONE
JUMP L1
```

L2 次の命令

i をGR5に置くことにする
Sを(ループ中は) GR3に置くことにする (変数Sに戻さない)

```
LD GR3, ZERO
ST GR5, S STはサボる
LD GR5, ZERO (GR5←0)
```

```
L1 CPA GR5, HYAKU (i-100)
JPL L2 (正→L2)
```

```
LD GR3, S LDはサボる
ADDA GR3, GR5 (S=S+i)
ST GR3, S STはサボる
```

```
ADDA GR5, ONE
JUMP L1
```

L2 ST GR3, S 脱出後 STする
次の命令



課題で追加した話は

ループの基本的な形は同じ

終了条件を念入りに検討する必要あり
特に = の時についてチェック

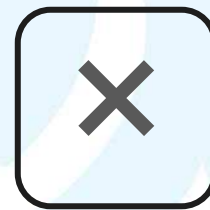
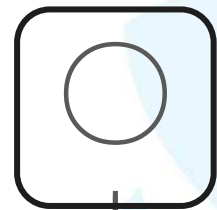
変数を毎回メモリに書き込まず、レジスタ
に置くことで、短縮可能
(コンパイラではこれを最適化と呼ぶ)



これは後から慣れればよい



ループ（for 文）の機械命令へ
変換の仕方がわかりましたか？



↓
次へ