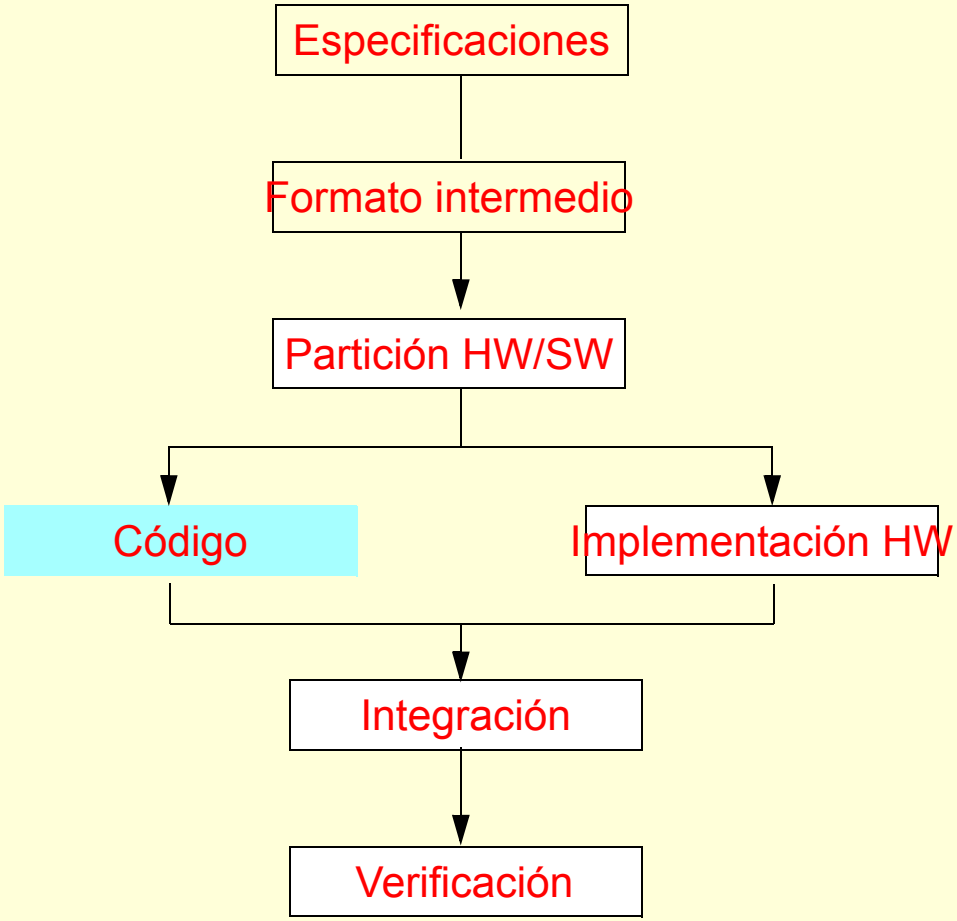
TEMA V: GENERACIÓN DE CÓDIGO

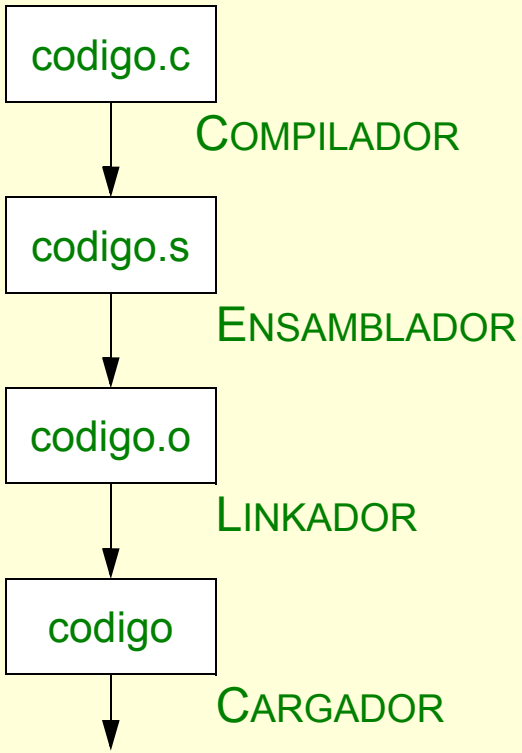


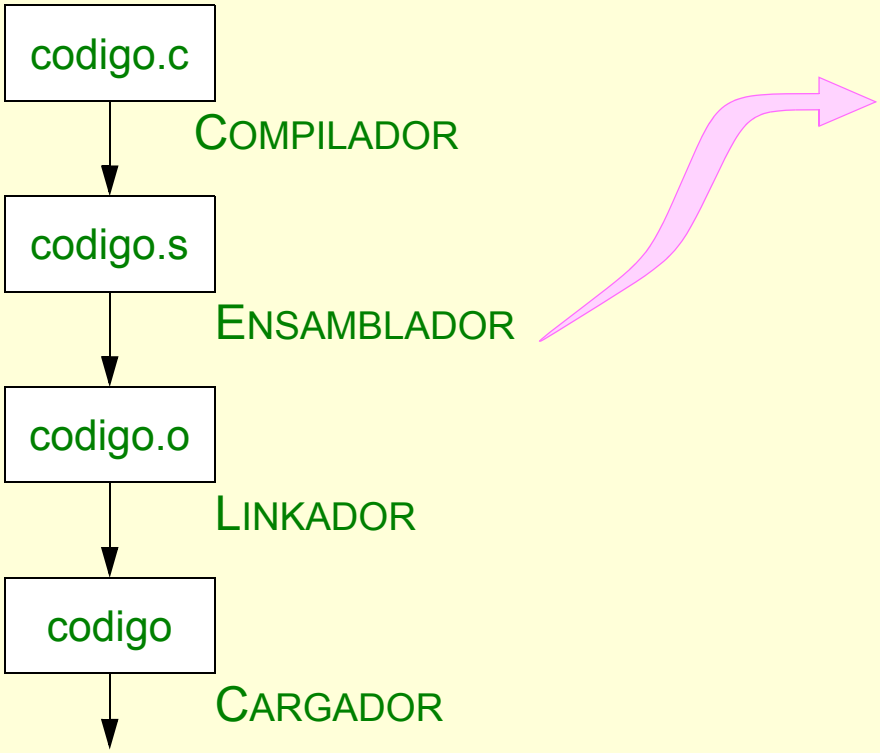
ARQUITECTURAS DEL SISTEMA SOFTWARE

- Conjunto de instrucciones
- Pipeline?
- Esquema de memoria
- Tipo de procesamiento

CRITERIOS DE OPTIMIZACIÓN

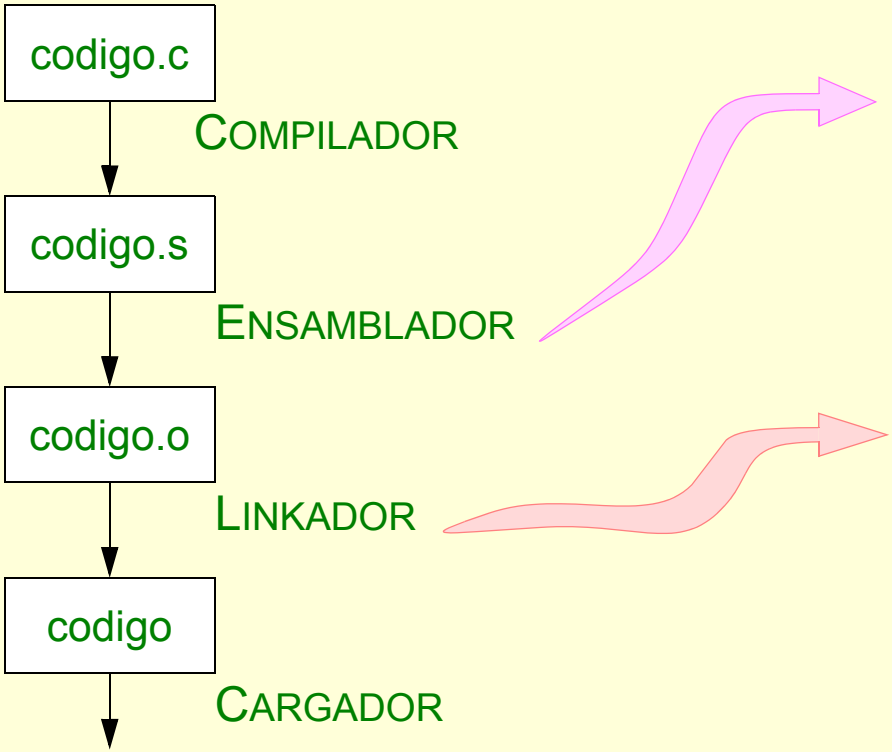
- Rendimiento
- Tamaño
- Consumo
- Utilización de recursos





PASAR DE CÓDIGO ENSAMBLADOR A CÓDIGO OBJETO

- Expansión de abreviaturas y macros
- Resolución de etiquetas locales
- Tabla de reubicación
- Obtención del formato máquina

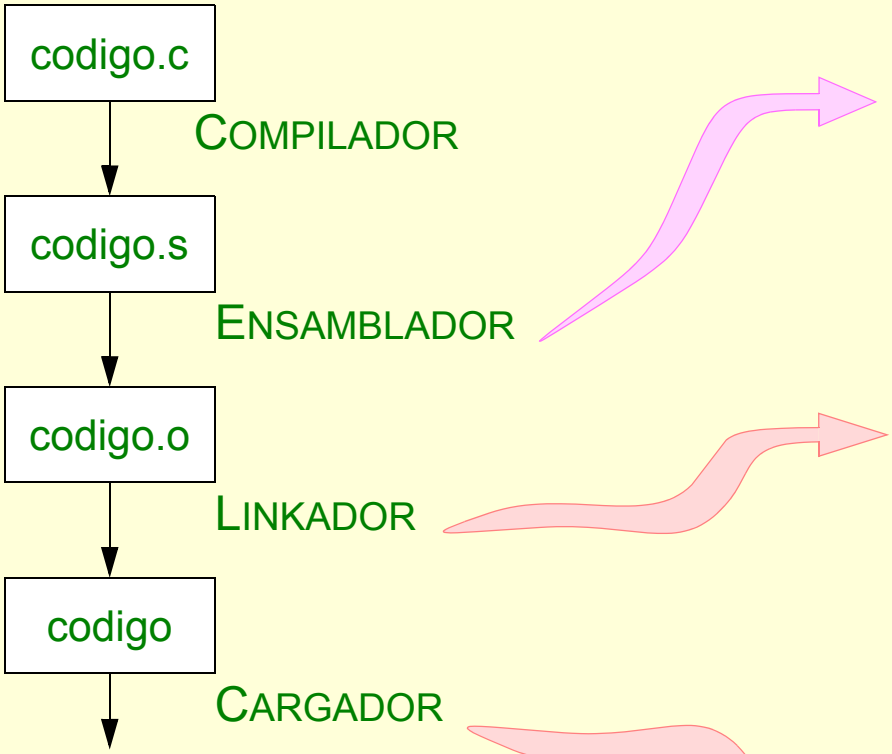


PASAR DE CÓDIGO ENSAMBLADOR A CÓDIGO OBJETO

- Expansión de abreviaturas y macros
- Resolución de etiquetas locales
- Tabla de reubicación
- Obtención del formato máquina

PASAR DE CÓDIGO OBJETO A CÓDIGO EJECUTABLE

- Leer todos los objetos y librerías utilizadas
- Combina las diferentes secciones
- Resuelve los símbolos globales
- Obtención del código ejecutable



PASAR DE CÓDIGO ENSAMBLADOR A CÓDIGO OBJETO

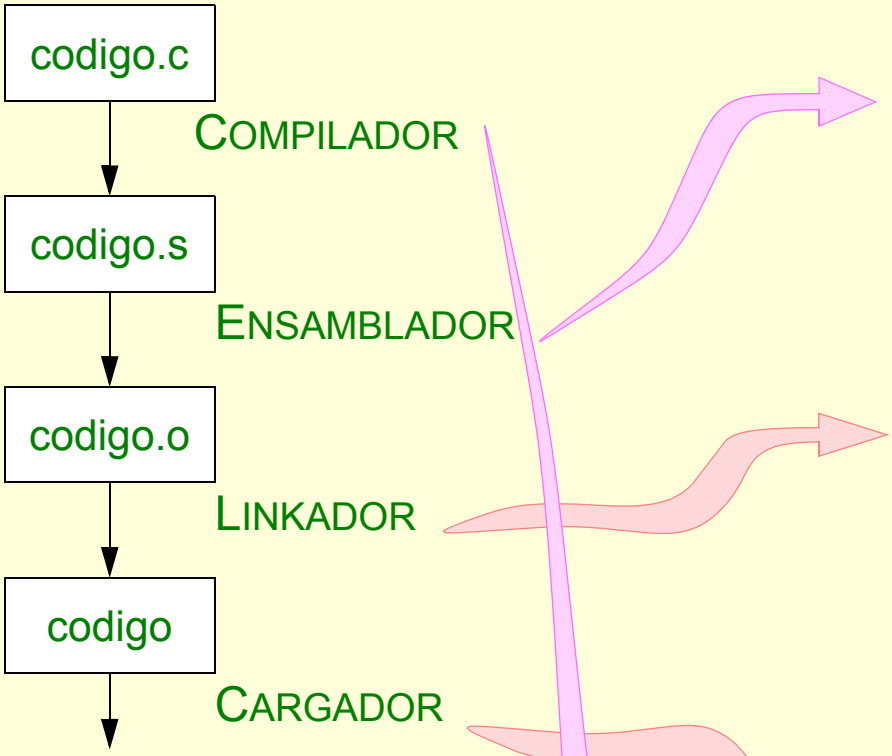
- Expansión de abreviaturas y macros
- Resolución de etiquetas locales
- Tabla de reubicación
- Obtención del formato máquina

PASAR DE CÓDIGO OBJETO A CÓDIGO EJECUTABLE

- Leer todos los objetos y librerías utilizadas
- Combina las diferentes secciones
- Resuelve los símbolos globales
- Obtención del código ejecutable

LLEVAR EL CÓDIGO EJECUTABLE A MEMORIA PRINCIPAL

- Leer todos los objetos y librerías utilizadas
- Combina las diferentes secciones
- Resuelve los símbolos globales
- Obtención del código ejecutable



PASAR DE CÓDIGO ENSAMBLADOR A CÓDIGO OBJETO

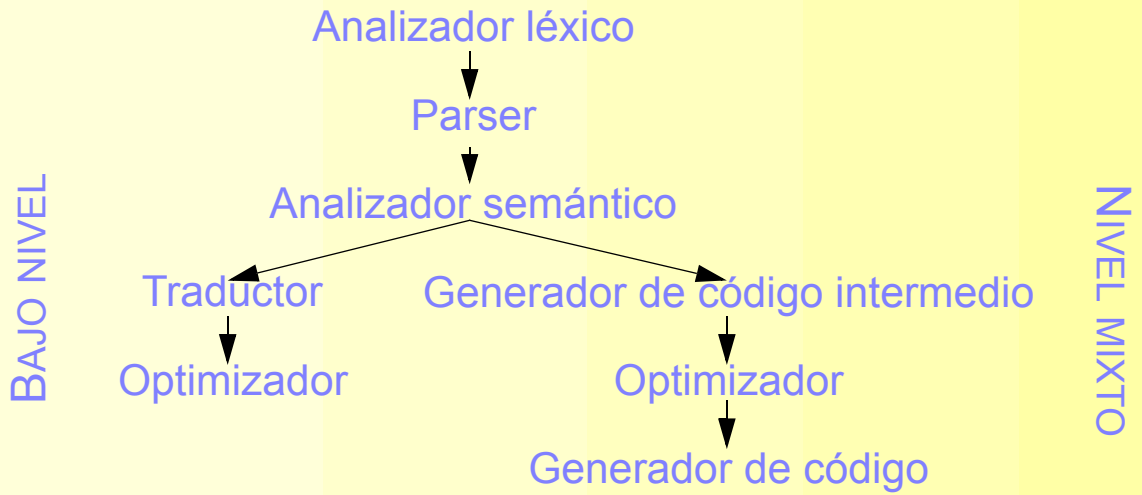
- Expansión de abreviaturas y macros
- Resolución de etiquetas locales
- Tabla de reubicación
- Obtención del formato máquina

PASAR DE CÓDIGO OBJETO A CÓDIGO EJECUTABLE

- Leer todos los objetos y librerías utilizadas
- Combina las diferentes secciones
- Resuelve los símbolos globales
- Obtención del código ejecutable

LLEVAR EL CÓDIGO EJECUTABLE A MEMORIA PRINCIPAL

- Leer todos los objetos y librerías utilizadas
- Combina las diferentes secciones
- Resuelve los símbolos globales
- Obtención del código ejecutable



ANÁLISIS

REDUCCIÓN DEL CONSUMO DE POTENCIA

Datapath de la ALU y unidades funcionales
Cache y sistemas de memoria
Buses del sistema
Circuitos de control y distribución más lógica de control

AUMENTO DE LA VELOCIDAD DE OPERACIÓN

Instrucciones existentes en el código
Número de accesos a memoria

REDUCCIÓN DEL ÁREA DE MEMORIA

Número de variables
Tipo de variables

PERMUTACIÓN DE BUCLES

```
for (i=0; i<N; i++)
    for (j=0; j<N; j++)
        P[j][i] = Q[j][i]
```

```
P[0][0] = Q[0][0]
P[1][0] = Q[1][0]

P[N][0] = Q[N][0]
P[0][1] = Q[0][1]
P[1][1] =Q[1][1]

P[N][1] = Q[N][1]
P[0][N] = Q[0][N]
P[1][N] = Q[1][N]

P[N][N] = Q[N][N]
```

Mayor número de accesos a memoria por falta de uso de la memoria cache

PEORES PRESTACIONES

```
for (j=0; j<N; j++)
    for (i=0; i<N; i++)
        P[j][i] = Q[j][i]
```

```
P[0][0] = Q[0][0]
P[0][1] = Q[0][1]

P[0][N] = Q[0][N]
P[1][0] = Q[1][0]
P[1][1] =Q[1][1]

P[N-1][N] = Q[N-1][N]
P[N][0] = Q[N][0]
P[N][1] = Q[N][1]

P[N][N] = Q[N][N]
```

Menor número de accesos a memoria por mayor uso de la memoria cache

MEJORES PRESTACIONES

REDUCCIÓN DEL TAMAÑO DE LA VENTANA DE LOS BUCLES

```
for (i=0; i<N; i++)
   for (j=0; j<N; j++)
      for (k=0; k<N; k++)
         X[i][j]=Y[i][j]+Z[k][j]
```

```
for (i1=0; i1<N; i1+=t)
   for (j1=0; j1<N; j1+=t)
      for (k1=0; k1<N; k1+=t)
         for (i=i1; i<min(i1+t,N); i++)
            for (j=j1; j<min(j1+t,N); j++)
               for (k=k1; k<min(k1+t,N); k++)
                  X[i][j]=Y[i][j]+Z[k][j]
```

Mayor número de accesos a memoria por falta de uso de la memoria cache

PEORES PRESTACIONES

Menor número de accesos a memoria por mayor uso de la memoria cache

MEJORES PRESTACIONES

FUSIÓN DE BUCLES

```
for (i=0; i<N; i++)
    P[i] = Q[i]
for (i=0; i<N; i++)
    P[i] = P[i]*R[i]
```

Mayor número de accesos a memoria por falta de uso de la memoria cache

PEORES PRESTACIONES

```
for (i=0; i<N; i++) {
    P[i] = Q[i]
    P[i] = P[i]*R[i]
}
```

Menor número de accesos a memoria por mayor uso de la memoria cache

MEJORES PRESTACIONES

DESDOBLE DE BUCLES

```
for (i=0; i<N; i++)
   for (j=0; j<N; j++)
      for (k=0; k<N; k++)
         X[i][j]=Y[i][j]+Z[k][j]
```

```
for (i=0; i<N; i+=b)
   for (j=0; j<N; j++)
      for (k=0; k<N; k++) {
         X[i][j]=Y[i][j]+Z[k][j]
         X[i+1][j]=Y[i][j]+Z[k][j]
         ...
         X[i+b-1][j]=Y[i][j]+Z[k][j]
}
```

Mayor número de accesos a memoria por falta de uso de la memoria cache

PEORES PRESTACIONES

Menor número de accesos a memoria por mayor uso de la memoria cache

MEJORES PRESTACIONES

REINDEXAR LA DIMENSIÓN

```
for (j=0; j<N; j++)
   for (i=0; i<N; i++)
      P[j][i] = Q[i][j]
```

```
P[0][0] = Q[0][0]
P[0][1] = Q[1][0]

P[0][N] = Q[N][0]
P[1][0] = Q[0][1]
P[1][1] = Q[1][1]

P[N-1][N] = Q[N][N-1]
P[N][0] = Q[0][N]
P[N][1] = Q[1][N]

P1[N][N] = Q[N][N]
```

Mayor número de accesos a memoria por falta de uso de la memoria cache

PEORES PRESTACIONES

```
for (j=0; j<N; j++)
   for (i=0; i<N; i++)
      P[j][i] = Q1[j][i]
```

```
P[0][0] = Q1[0][0]
P[0][1] = Q1[0][1]

P[0][N] = Q1[0][N]
P[1][0] = Q1[1][0]
P[1][1] = Q1[1][1]

P[N-1][N] = Q1[N-1][N]
P[N][0] = Q1[N][0]
P[N][1] = Q1[N][1]

P1[N][N] = Q1[N][N]
```

Menor número de accesos a memoria por mayor uso de la memoria cache

MEJORES PRESTACIONES

MINIMIZAR ACCESOS A MEMORIA

```
for (i=0; i<N; i++)
    for (j=0; j<N; j++)
        P[i] = ...
...
}
```

Mayor número de accesos a memoria por falta de uso de la memoria cache

PEORES PRESTACIONES

```
for (i=0; i<N; i++) {
    P[i] = ...
    for (j=0; j<N; j++)
...
}
```

Menor número de accesos a memoria por mayor uso de la memoria cache

MEJORES PRESTACIONES

REDUCCIÓN DEL ÁREA DE MEMORIA

```
for (i=0; i<N; i++) {
    ... = P[i]
    for (j=0; j<N; j++)
        P[i] = R[i]
}
```

Mayor número de accesos a memoria por falta de uso de la memoria cache

PEORES PRESTACIONES

```
for (i=0; i<N; i++) {
    ... = P[i]
    for (j=0; j<N; j++)
        Q[i] = R[i]
}
```

Menor número de accesos a memoria por mayor uso de la memoria cache

MEJORES PRESTACIONES