

# ПРОЈЕКТОВАЊЕ МАКРОАСЕМБЛЕРА

- ◆ Макроасемблер
- ◆ Потребна проширења асемблера

# МАКРОАСЕМБЛЕР

- ◆ Макроасемблер преводи полазни програм написан на макроасемблерском језику у извршиви машински програм.
- ◆ Приликом израде асемблерског програма често се делови програма понављају више пута.
- ◆ Уводи се отворени потпрограм, који се назива МАКРО ДЕФИНИЦИЈА, а позива се МАКРО ИНСТРУКЦИЈОМ.

# МАКРОАСЕМБЛЕРСКИ ЈЕЗИК

## ◆ Особине

- Језик вишег нивоа
  - ◆ Уз задржавање предности асемблера
  - ◆ Могућност стандардизације
  - ◆ Третира се као проширење асемблера

## ◆ Примене

- Описивање оперативног система (ОС)
  - ◆ Генерисање ОС макро претпроцесирањем
- Симболичке машинске инструкције нове апстрактне машине, нпр. `save`, `restore`, итд.

# МАКРОИ У ВИШИМ ПРОГРАМСКИМ ЈЕЗИЦИМА

- ◆ Макро искази (макрои) постоје и у вишим програмским језицима
- ◆ Нпр. у језику Це(++)
  - Директиве Це макро претпроцесора
    - ◆ #include, #ifdef, #endif, itd.
    - ◆ Макрои се дефинишу помоћу #define
      - Нпр. #define MAX(A,B) ((A)>(B): (A) ? (B))
      - Макро MAX(x,y) се развија у ((x)>(y)? (x) : (y))

# Проширења асемблера ради обраде макроинструкција:

- ◆ Прво: да препозна макроинструкцију.
- ◆ Друго: да прошири макроинструкцију.
- ◆ Да би то урадио он најпре треба да пронађе и сачува макродефиницију која одговара макроинструкцији.
- ◆ Макро дефиниција започиње MACRO, а завршава са ENDM псеудо инструкцијом.

# Препознавање макроинструкција:

- ◆ Препознавањем псеудо операције MACRO – име је у пољу лабеле
- ◆ Најпростији начин: додати врсту у табели кода операције.
  - Ова врста садржи назив макро инструкције и указивач на одговарајућу макро дефиницију.
- ◆ Дефиниције макроа се чувају у ТАБЕЛИ МАКРО ДЕФИНИЦИЈА.

# Обрада макро дефиниције

- ◆ Врста табеле КОП садржи
  - симболички КОП, његов тип, одговарајући нумерички код и тип његових операнда
  - Макро нема нумерички код
    - ◆ Ту се смешта индекс одговарајуће макродефиниције у табели макро дефиниција (ТМД)
    - ◆ Задњи елеменат, тип операнда, није битан за макро инструкцију

# Пример ТМД

- ◆ Смешта се цела макро дефиниције
  - Све симболичке инструкције унутар ње су у редоследу у коме су се појавили у процедури полазног језика

INDEKS	TABELA MAKRO DEFINICIJE (TMD)
15	ADDUP MACRO P1, P2, P3 ADD AX, P1 ADD BX, P2 ADD CX, P3 MEND



# ЛИСТА НАЗИВА ПАРАМЕТАРА:

- ◆ Поред табеле макро дефиниција користи се и листа назива параметара (ЛНП).
- ◆ Она успоставља везу између фиктивних и стварних параметара макро инструкције.
- ◆ Унутар макро дефиниције параметри се могу референцирати и преко индекса у листи параметара.

# Пример ЛНП

◆ Нпр. макроинструкција:

- ADDUP DATA1, DATA2, DATA3
  - ◆ ADDUP назив макроинструкције
  - ◆ DATA1, DATA2, DATA3 су стварни параметри („аргументи“)

**TABELA LISTE NAZIVA PARAMETARA (LNP)**

INDEKS	STVARNI PARAMETAR	FIKTIVNI PARAMETAR
1	DATA 1	P1
2	DATA 2	P2
3	DATA 3	P3

# Проширивање макро инструкција (обавља се у фази превођења):

- ◆ Макро инструкција се замењује телом одговарајуће макро дефиниције.
- ◆ Формални параметар у телу макро дефиниције се замењује одговарајућом вредношћу стварног параметра из поља операнда макро инструкције која се проширује.
- ◆ Даље следи први пролаз асемблера.

# Специфичности проширивања макроинструкција:

- ◆ Макро језик укључује макро променљиве, уграђене функције за баратање низовима знакова, изразе, и исказе доделе и избора.
- ◆ Дозвољена је само и међу рекурзија – опасност од бесконачног развоја.
- ◆ Резултат – променљив број инструкција.
- ◆ Замена фиктивних параметара стварним се обавља на нивоу ASCII знакова.

# Пример програма у ЛПРС асемблеру

```
#define NOP and R0, R0, R0  
#define ZERO(x) sub x, x, x
```

```
.text
```

```
Begin: // program racuna a * b
```

```
    ZERO(R0)
```

```
    ld R1, a
```

```
    ld R2, b
```

```
Loop:
```

```
    add R0, R0, R1
```

```
    dec R2, R2
```

```
    jmpnz Loop
```

```
    NOP
```