AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Implementacja maszyny wirtualnej dla funkcyjnych języków programowania wspierających przetwarzanie współbieżne.

Kajetan Rzepecki

Wydział EAIiIB
Katedra Informatyki Stosowanej

20 grudnia 2013

Wprowadzenie

Maszyna wirtualna - środowisko uruchomieniowe języków programowania uniezależniające je od platformy sprzętowej.

Wprowadzenie

Maszyna wirtualna - środowisko uruchomieniowe języków programowania uniezależniające je od platformy sprzętowej.

W skład pracy wchodzą:

- Implementacja interpretera kodu bajtowego.

Wprowadzenie

Maszyna wirtualna - środowisko uruchomieniowe języków programowania uniezależniające je od platformy sprzętowej.

W skład pracy wchodzą:

- Implementacja interpretera kodu bajtowego.
- Implementacja kolektora obiektów nieosiągalnych.

Wprowadzenie

Maszyna wirtualna - środowisko uruchomieniowe języków programowania uniezależniające je od platformy sprzętowej.

W skład pracy wchodzą:

- Implementacja interpretera kodu bajtowego.
- Implementacja kolektora obiektów nieosiągalnych.
- Implementacja Modelu Aktorowego (ang. Actor Model).

Wprowadzenie - Motywacja pracy

```
transmogrify(Data) ->
    Pids = framework:spawn_bajilion_procs(fun do_stuff/1),
    JSON = json:decode(Data),
    framework:map_reduce(Pids, JSON). %% $#@~^@
```

Wprowadzenie - Motywacja pracy

```
transmogrify(Data) ->
    Pids = framework:spawn_bajilion_procs(fun do_stuff/1),
    JSON = json:decode(Data),
    framework:map_reduce(Pids, JSON). %% $#@~^@
```

- Mniejsza logika przepływu danych.

Wprowadzenie - Motywacja pracy

```
transmogrify(Data) ->
    Pids = framework:spawn_bajilion_procs(fun do_stuff/1),
    JSON = json:decode(Data),
    framework:map_reduce(Pids, JSON). %% $#@~@
```

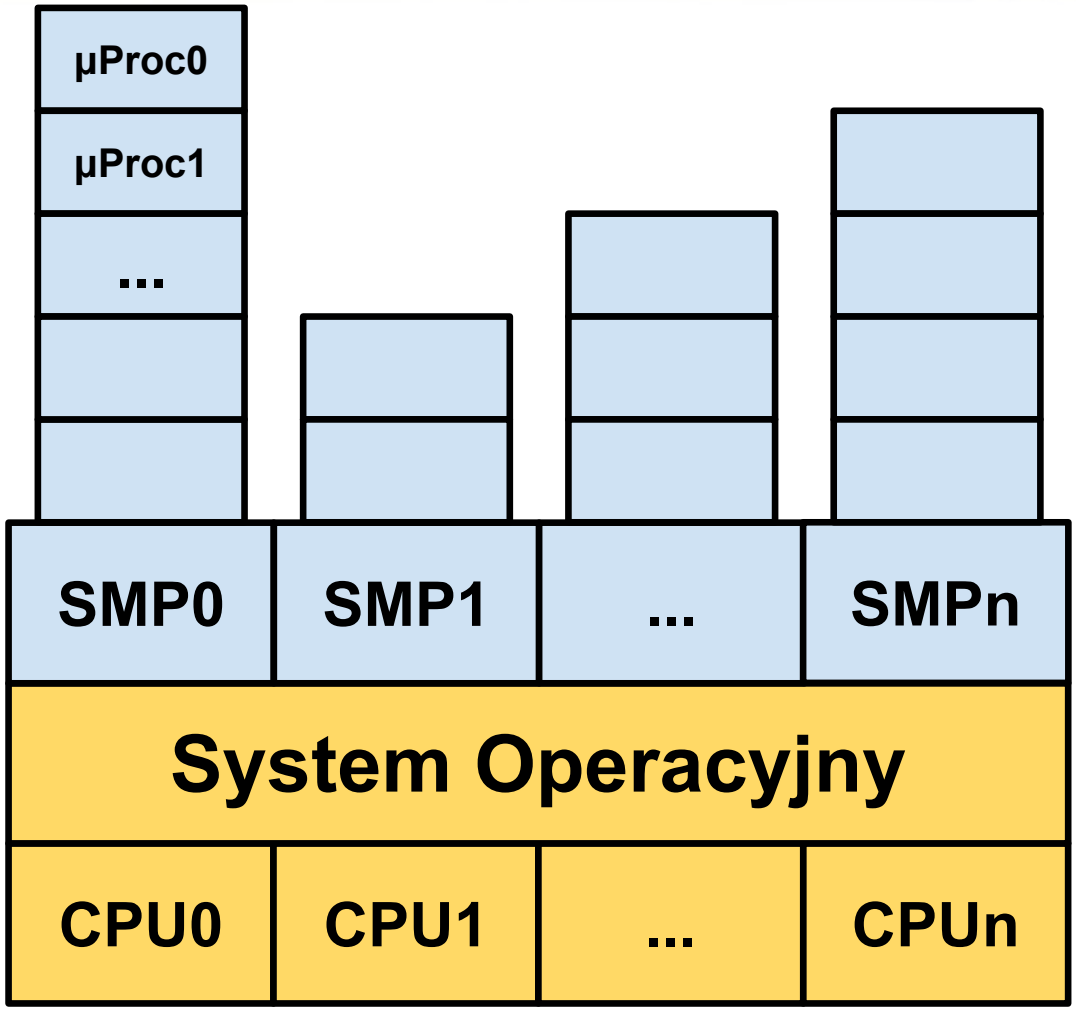
- Mniejsza logika przepływu danych.
- Zwielokrotnienie parsowania pliku JSON.

Wprowadzenie - Motywacja pracy

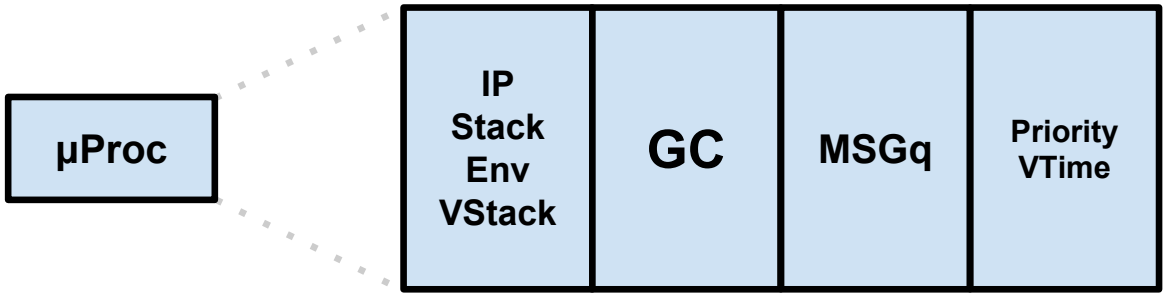
```
transmogrify(Data) ->
    Pids = framework:spawn_bajilion_procs(fun do_stuff/1),
    JSON = json:decode(Data),
    framework:map_reduce(Pids, JSON). %% $#@~^@
```

- Mniejsza logika przepływu danych.
- Zwielokrotnienie parsowania pliku JSON.
- Działa szybciej. (!?)

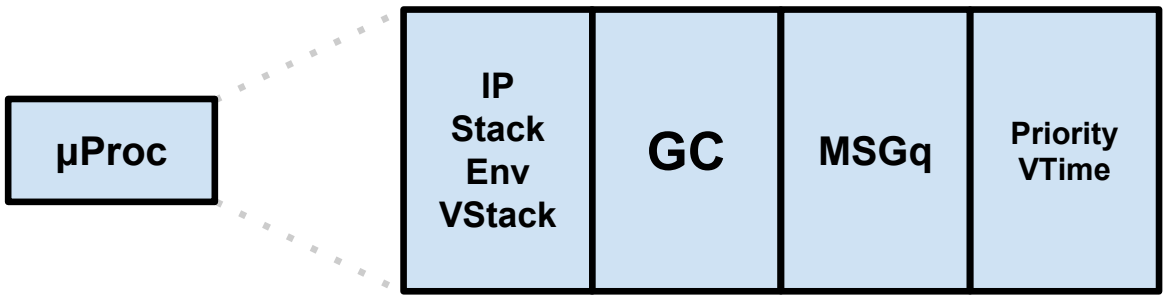
ThesisVM - Architektura



ThesisVM - Interpreter kodu bajtowego



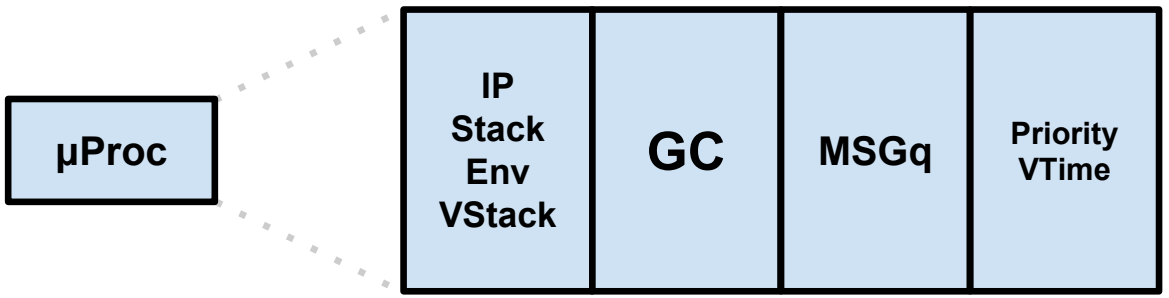
ThesisVM - Interpreter kodu bajtowego



Oparty o **Three Instruction Machine**:

- Niewielka ilość rejestrów.

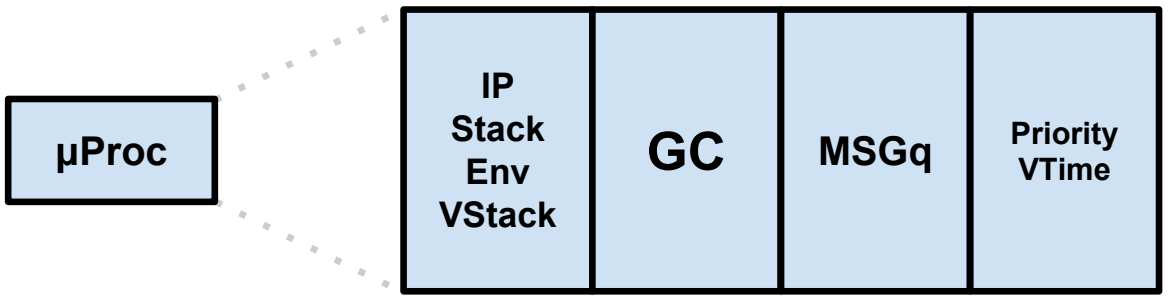
ThesisVM - Interpreter kodu bajtowego



Oparty o **Three Instruction Machine**:

- Niewielka ilość rejestrów.
- Niewielka ilość instrukcji.

ThesisVM - Interpreter kodu bajtowego



Oparty o **Three Instruction Machine**:

- Niewielka ilość rejestrów.
- Niewielka ilość instrukcji.
- Architektura **CISC**.

ThesisVM - Przykład kodu bajtowego

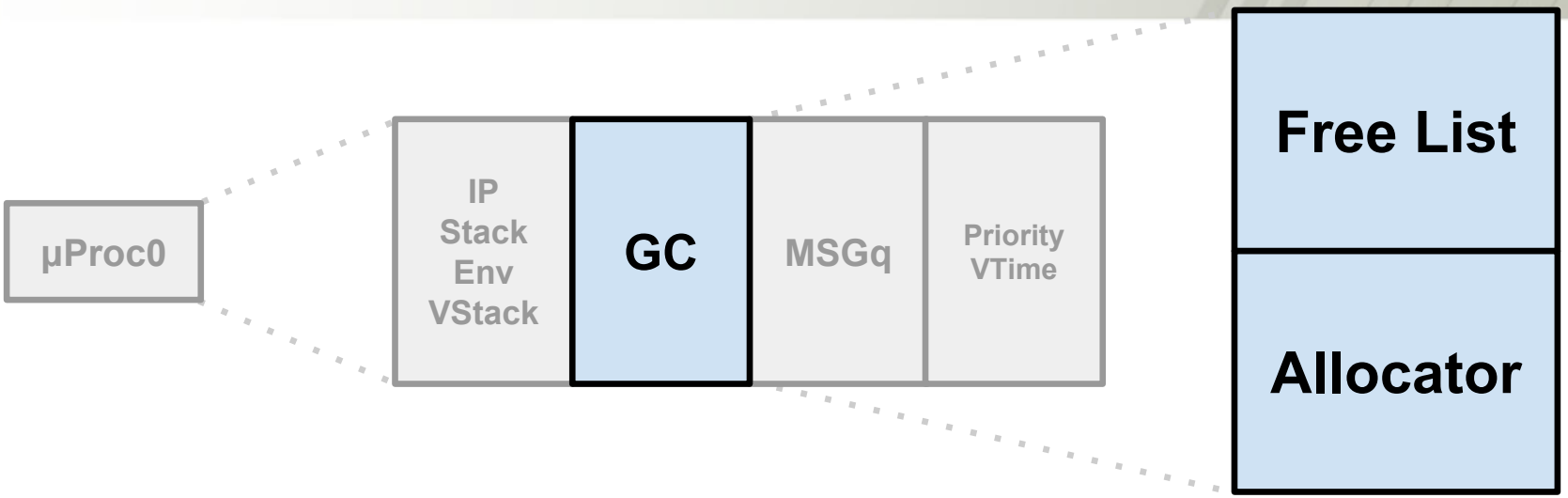
```
(define (add a b)
  (+ a b))
(add 2 2)
```

ThesisVM - Przykład kodu bajtowego

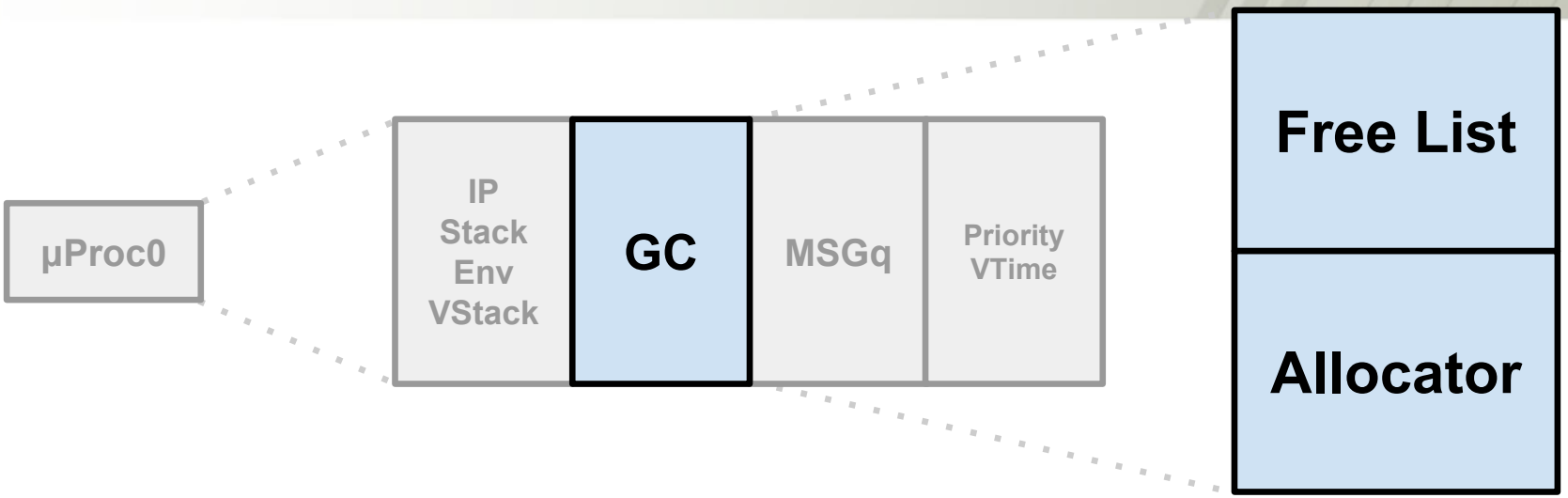
```
(define (add a b)
  (+ a b))
(add 2 2)
```

```
__start: PUSH 2        # Połóż "a" na stosie.
         PUSH 2
         ENTER add     # Wejdź do domknięcia "add".
add:     TAKE 2        # Pobierz dwa argumenty ze stosu.
         PUSH __add0
         ENTER 1       # PUSHC 2, RETURN
__add0:  PUSH __add1
         ENTER 0
__add1:  OP_ADD
         RETURN
```

ThesisVM - Kolekcja obiektów nieosiągalnych

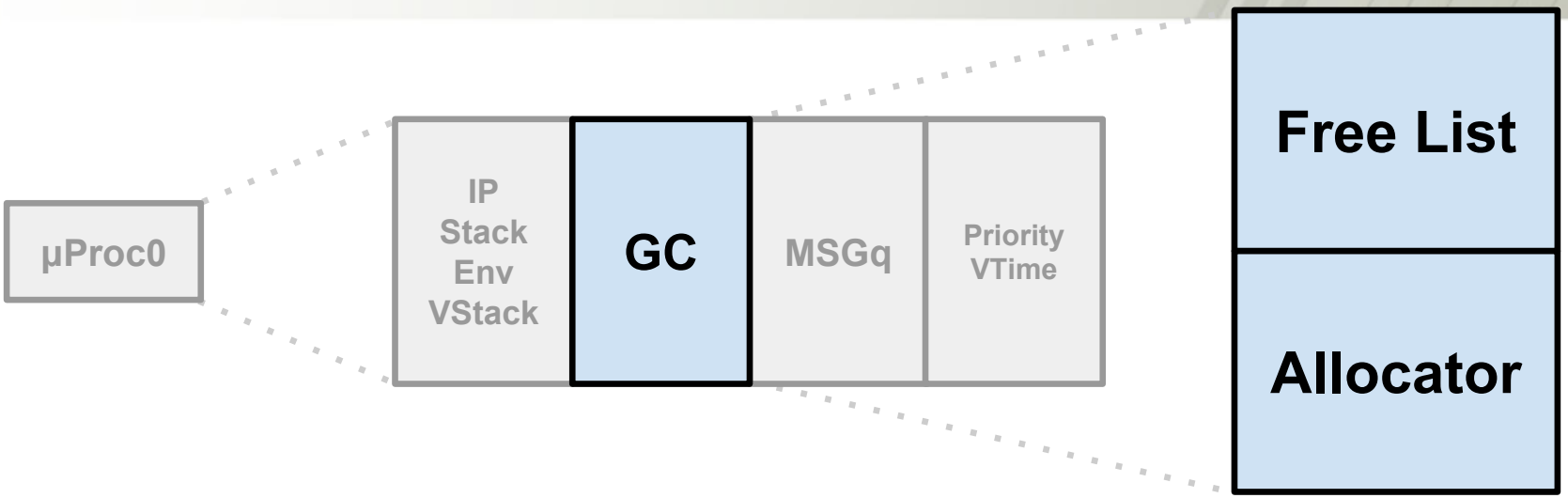


ThesisVM - Kolekcja obiektów nieosiągalnych



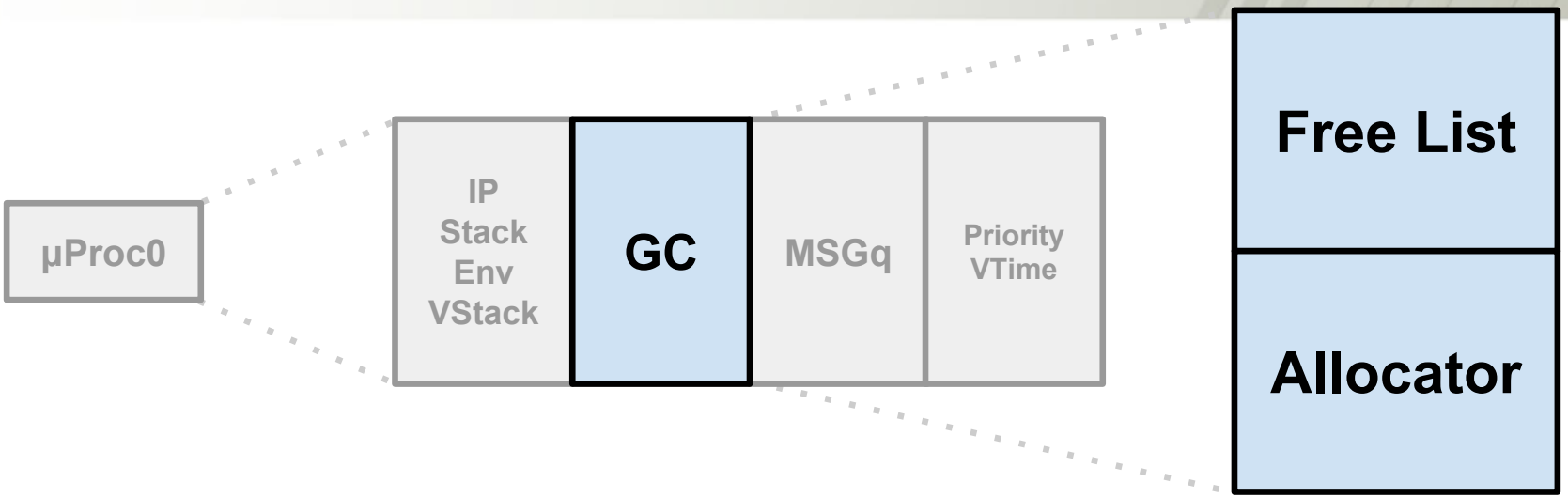
- Wykorzystuje leniwe zliczanie referencji.

ThesisVM - Kolekcja obiektów nieosiągalnych



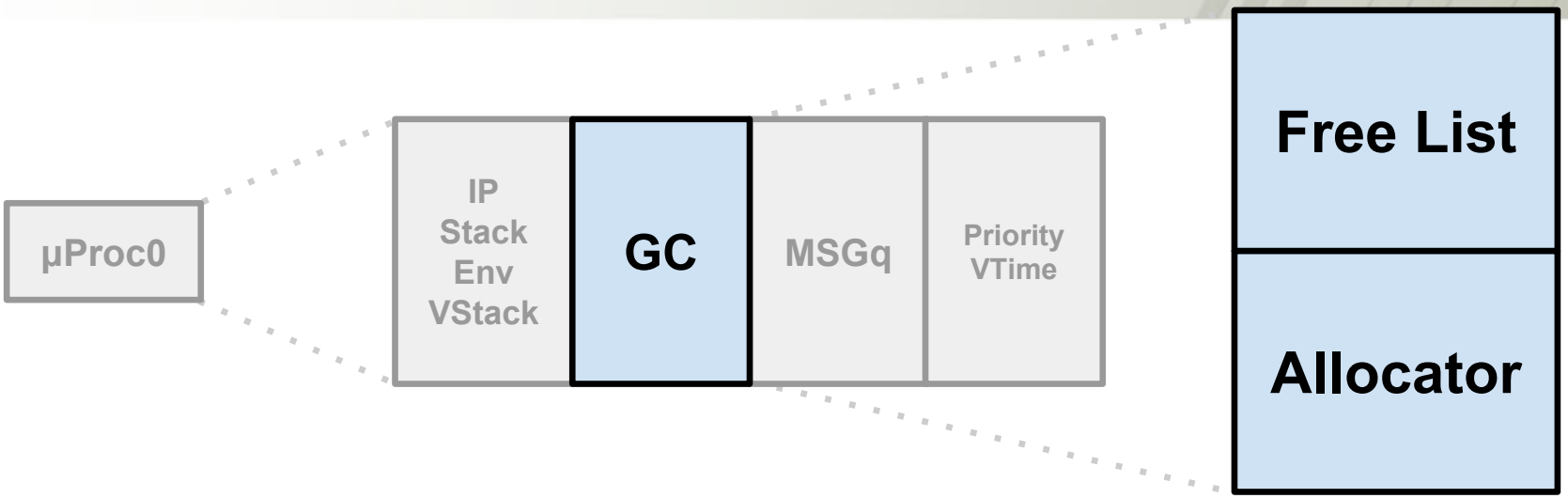
- Wykorzystuje leniwe zliczanie referencji.
- Kolekcja pamięci procesu nie zależy od innych procesów.

ThesisVM - Kolekcja obiektów nieosiągalnych



- Wykorzystuje leniwe zliczanie referencji.
- Kolekcja pamięci procesu nie zależy od innych procesów.
- "Ostatni gasi światło."

ThesisVM - Kolekcja obiektów nieosiągalnych



- Wykorzystuje leniwe zliczanie referencji.
- Kolekcja pamięci procesu nie zależy od innych procesów.
- "Ostatni gasi światło."
- Proste obiekty ($\leq$ 8 bajtów) są kopiowane.

ThesisVM - Leniwe zliczanie referencji

- Opóźnienie kolekcji obiektu do następnej alokacji.

ThesisVM - Leniwe zliczanie referencji

- Opóźnienie kolekcji obiektu do następnej alokacji.
- Szybkie dealokacje.

ThesisVM - Leniwe zliczanie referencji

- Opóźnienie kolekcji obiektu do następnej alokacji.
- Szybkie dealokacje.
- Szybkie alokacje zamortyzowane listą wolnych obiektów.

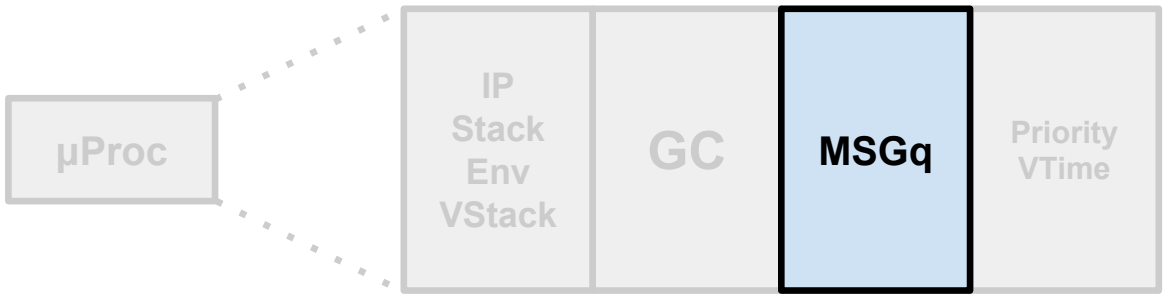
ThesisVM - Leniwe zliczanie referencji

- Opóźnienie kolekcji obiektu do następnej alokacji.
- Szybkie dealokacje.
- Szybkie alokacje zamortyzowane listą wolnych obiektów.
- Pamięć nie jest natychmiastowo zwracana do Systemu Operacyjnego.

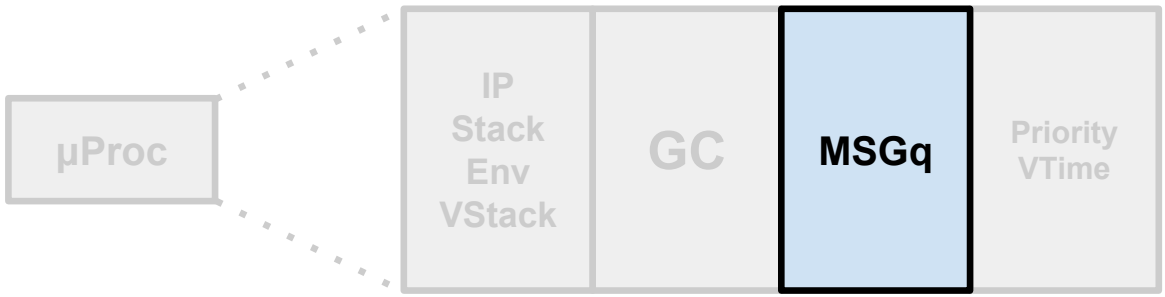
ThesisVM - Leniwe zliczanie referencji

- Opóźnienie kolekcji obiektu do następnej alokacji.
- Szybkie dealokacje.
- Szybkie alokacje zamortyzowane listą wolnych obiektów.
- Pamięć nie jest natychmiastowo zwracana do Systemu Operacyjnego.
- Wymaga atomowych operacji na liczniku referencji oraz barier pamięci.

ThesisVM - Model Aktorowy

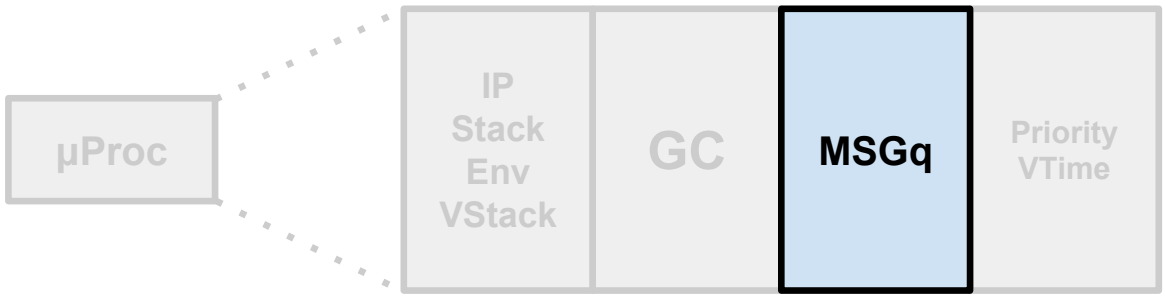


ThesisVM - Model Aktorowy



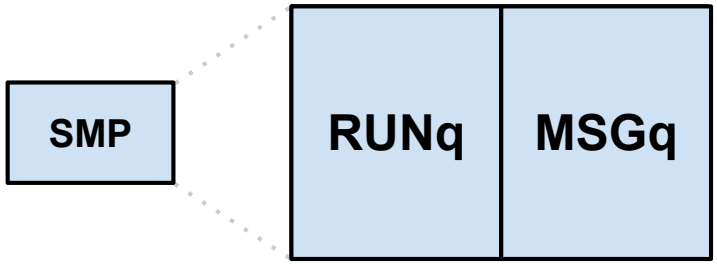
- Idetyfikator procesu (pid) to wskaźnik na kontekst procesu.

ThesisVM - Model Aktorowy

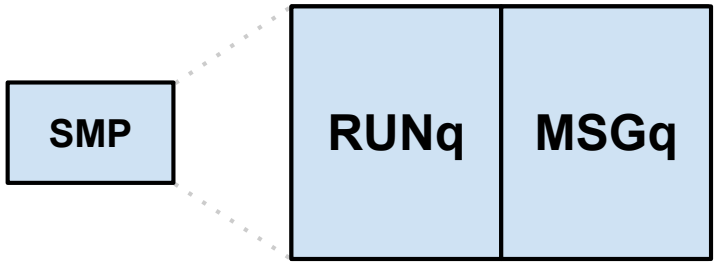


- Idetyfikator procesu (pid) to wskaźnik na kontekst procesu.
- Wykorzystuje kolejki nieblokujące.

ThesisVM - Symetryczny Multiprocessing

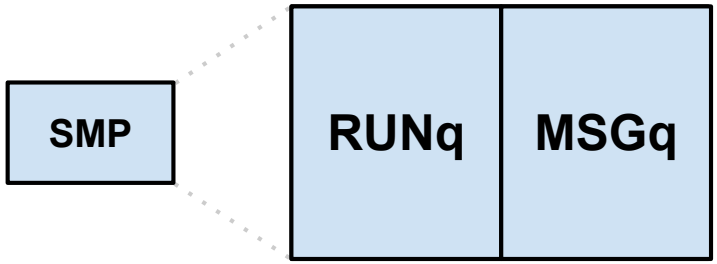


ThesisVM - Symetryczny Multiprocessing

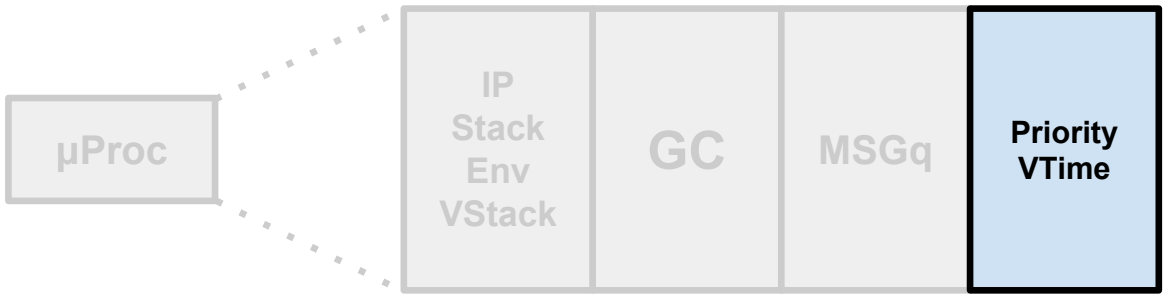


- Wykorzystuje Model Aktorowy!

ThesisVM - Symetryczny Multiprocessing



- Wykorzystuje Model Aktorowy!



- Procesy są wywłaszczane (ang. preemptive concurrency).

Podsumowanie

Projekt implementuje:

- Interpreter kodu bajtowego oparty o **Three Instruction Machine**.

Podsumowanie

Projekt implementuje:

- Interpreter kodu bajtowego oparty o **Three Instruction Machine**.
- Kompilator kodu bajtowego.

Podsumowanie

Projekt implementuje:

- Interpreter kodu bajtowego oparty o **Three Instruction Machine**.
- Kompilator kodu bajtowego.
- Kolektor obiektów nieosiągalnych oparty o **leniwe zliczanie referencji**.

Podsumowanie

Projekt implementuje:

- Interpreter kodu bajtowego oparty o **Three Instruction Machine**.
- Kompilator kodu bajtowego.
- Kolektor obiektów nieosiągalnych oparty o **leniwe zliczanie referencji**.
- Architekturę SMP oraz Model Aktorowy oparty o **kolejki nieblokujące**.

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Dziękuję za uwagę.

Kajetan Rzepecki